



NRL/MR/7330--10-9238

## Validation Test Report for LAGER 1.0

MICHAEL CARNES

*Ocean Sciences Branch  
Oceanography Division*

PATRICK HOGAN

*Ocean Dynamics and Prediction Branch  
Oceanography Division*

April 13, 2010

Approved for public release; distribution is unlimited.



## Contents

1	Introduction.....	1
2	Data Set.....	2
3	Discussion of typical CTD errors associated with gliders.....	5
3.1	Seagliders.....	5
3.2	Slocums.....	7
4	Validation.....	8
5	Conclusions.....	15
6	References.....	16
7	APPENDIX A.....	17

# 1 Introduction

LAGER v1.0 (Local Automated Glider Editing Routine) is the first version of an ocean glider quality control system developed for the Naval Oceanographic Office (NAVO) Glider Operations Center (GOC). The LAGER v1.0 methods and operation are fully documented in the LAGER Manual (Carnes, 2008), also included here in Appendix A. It was developed to import raw data returned from ocean gliders operated by NAVO, to detect and flag erroneous observations, and to reformat data and flags into a format accepted by the NAVO Real-Time Data Handling System (RTDHS). The software for this system, which is coded in Matlab, performs four main steps. The first step imports unedited raw data and metadata from either Slocum gliders or Seagliders transmitted to NAVO in real time over the Iridium Satellite LLC data communications system. When received, the data is reformatted into a Netcdf format similar to the output from the Seaglider Base Station system. The second step performs an automated quality assessment of the glider observations which detects and flags erroneous observations. The second step also applies calibration algorithms to optical observations, when required, and extracts, checks, and formats various auxiliary data such as GPS positions. At the end of this step, flags are set indicating whether the resulting file of data requires further evaluation in the third step of LAGER. If further evaluation is not required, the file is sent immediately to the fourth step. The third step displays glider observations in a GUI-based manual quality control program operated by personnel of the NAVO GOC. The operator displays profile observations from each file. If necessary, the operator changes quality control (QC) flags and edits observations using the controls available in the GUI. The fourth step of LAGER reformats the file of edited and flagged glider observations into the BUFR file format required by the RTDHS.

The primary goal of LAGER is to detect and flag all erroneous observations prior to being passed to the RTDHS. Some types of observational errors returned by the gliders are difficult to detect and properly flag by the present quality control algorithms incorporated in LAGER. For this reasons, the automated quality control software also includes algorithms that attempt to identify possible failure of the QC procedures based on the number of observations flagged as bad or on the types of errors detected. When a possible failure has been detected, the file is marked with flags indicating that manual quality control is required. The second goal of LAGER is to minimize the number of files being sent to the manual quality control inspection step, while at the same time enforcing the primary goal. We know that the automated QC algorithms will not detect every error. However, the file should not be sent to the manual QC step unless there is reasonable evidence that errors still remain in the file that were not detected or properly flagged by the automated QC. This second goal is required in order to minimize the amount of time spent by GOC personnel performing manual data QC.

The LAGER automated QC sends a combined temperature-and-salinity profile for further evaluation in the manual QC step if any of the following conditions are true:

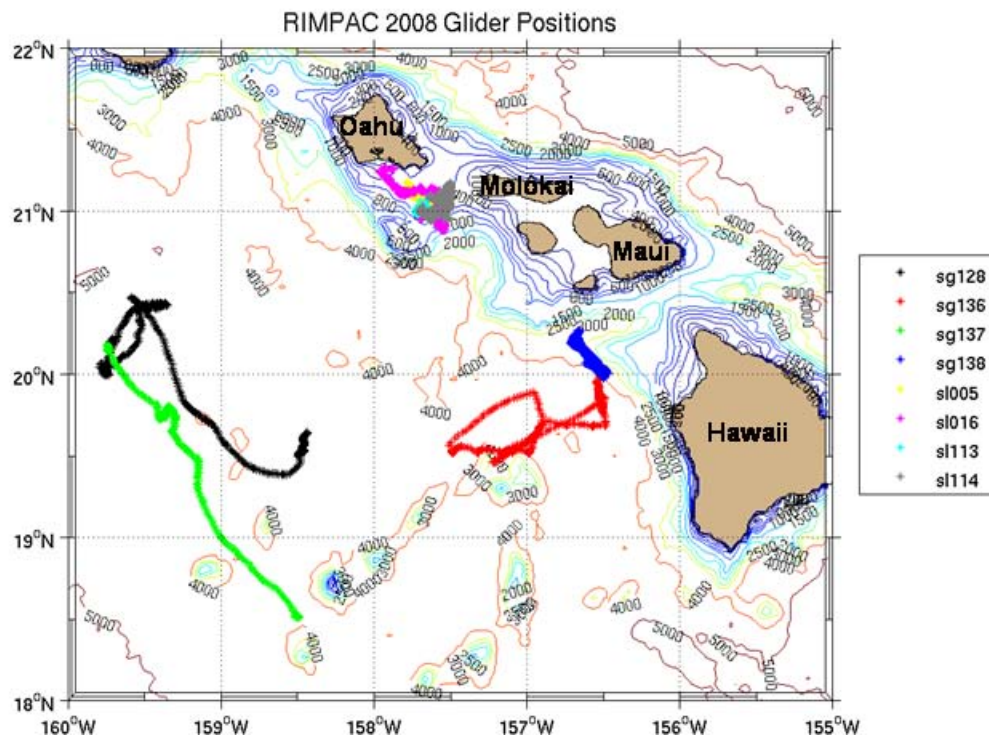
1. The full temperature or salinity profile is flagged for deletion. For either the temperature or salinity, the full profile is flagged for deletion if the number of observations flagged as bad is greater than 30 or greater than the 30% of the total number of observations. In this case and in cases 2 and 3 below, the number of observations counted as bad does not include those that are always flagged as bad within a specified distance from the surface (2 m for Seaglidors and 0.2 m for Slocums). It also does not include observations flagged as bad within a specified depth range of the top or bottom when the vertical velocity is below a specified range. The full details are discussed in the LAGER Manual for Version 1.0.
2. The number of temperature observations flagged as bad is greater than 15 or greater than 15% of the total number of observations.
3. The number of salinity observations flagged as bad is greater than 15 or greater than 15% of the total number of observations.
4. The number of adjacent observation pairs determined to be statically unstable, prior to application of the stabilization algorithm, is greater than 15 or greater than 15% of the total number of adjacent observation pairs.
5. Any temperature or salinity observation is more than 5 GDEM standard deviations from the GDEM monthly mean.
6. For only the Slocum glider, there are one or more gaps of at least 10 m depth with no observations or there are three or more gaps of at least 6 m depth with no observations.

This report documents a validation of the automated portion of LAGER v1.0 using temperature and salinity observations from four Slocums and four Seaglidors operated by NAVO during the RIMPAC08 exercise conducted near Hawaii in July 2008. During the exercise, LAGER was not used operationally by the GOC, but it was run offline by GOC personnel to test its performance against the procedures already in place at NAVO. The validation is primarily intended to demonstrate that the LAGER scheme for deciding whether to send a profile for further review by manual QC performed adequately. The scheme works well if it sends profiles to manual QC only when there are unflagged errors remaining in the file after the automated QC.

## **2 Data Set**

RIMPAC (Rim of the Pacific) is a multi-national exercise scheduled biennially by the U.S. Pacific Fleet. The 2008 exercise included ten participating countries and took place in the Hawaiian operating area from June 29 to July 31, 2008. Four Seaglidors owned and operated by NAVO were deployed from the USNS SUMNER for anti-submarine warfare operations and two shallow-water Slocums owned and operated by NAVO and two Slocums provided by Rutgers University's Coastal Ocean Observation Laboratory (RUCOOL) were deployed for mine warfare operations. Each glider was equipped with a Sea-Bird Electronics 41cp (non-pumped) CTD and various optical sensors. Only the temperature, salinity, and pressure measured by the CTDs are used in the validation study of this report. The operational control of the gliders and the processing of observations were centered at the NAVO GOC.

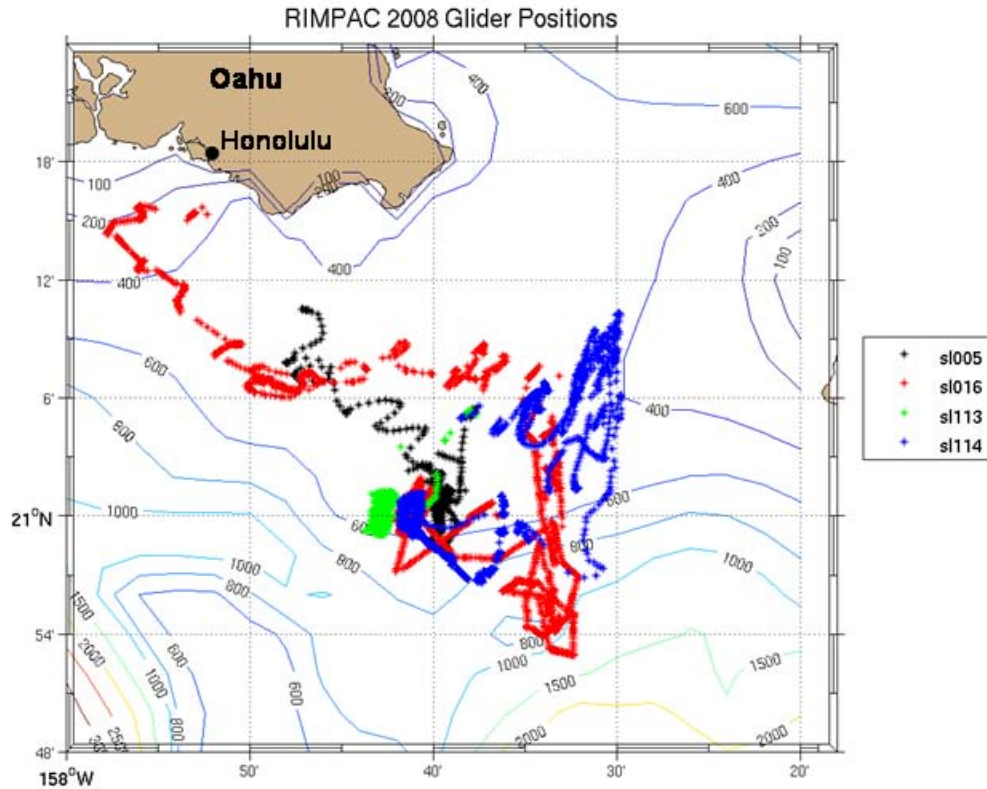
The path of each of the eight gliders is shown on the map in Figure 1. The Seagliders maintained positions over deep water west of the line of islands between Hawaii and Oahu. The Seagliders, named sg136, sg137, and sg138, primarily surveyed the water between the surface and about 1000 m depth, but sg128 surveyed between the surface and only about 650 m depth. The path of the four slocums, also shown in the smaller map east of Molokai and south of Oahu, primarily crossed back and forth over the Penguin Bank, a submerged coral-capped volcano with minimum water depths near about 50 m. The Slocums, named sl005, sl016, sl113, and sl114 mainly surveyed between the surface and the bottom over water about 55 m deep, but occasionally surveyed down the sides of the bank down to 95 m depth for sl005 and sl016 and to about 190 m depth for sl113 and sl115. The bottom depths (in units of meters) shown contoured in Figures 1 and 2, were extracted from the global 2-minute-resolution DBDB2 v3.0 bathymetry ([http://www7320.nrlssc.navy.mil/DBDB2\\_WWW](http://www7320.nrlssc.navy.mil/DBDB2_WWW)) developed by D. S. Ko at the Naval Research Laboratory (NRL) (Marks and Smith (2006)). The bottom depths shown in the area under the glider tracks are between 400 m and 1000 m, much deeper than shallow depths of the Penguin Bank



**Figure 1** Positions occupied by the four Seagliders and four Slocum gliders during the RIMPAC 2008 exercise. Bottom depths are in meters.

All of the glider observation files were processed during the exercise through two separate paths, the operational processing system and the offline experimental LAGER processing system. The operational processing employed NOPEDS, a Matlab GUI-based profile editor used to display and manually edit XBT, CTD, and glider CTD profiles) plus extra temporary software to process the raw Slocum data which was installed by John

Kerfoot from Rutgers University's RUCOOL prior to the exercise . The raw Slocum data processed by this system was reformatted in a format compatible with NOPEDS, and then the files were manually edited in NOPEDS. Both processing systems produced a final quality-controlled data set of glider observations collected during the exercise. The statistics derived from a comparison of the final data sets from these two systems comprises the LAGER validation results.



**Figure 2** Positions of the four Slocum gliders, primarily over the Penguin Bank, during the RIMPAC 2008 exercise. Bottom depths are in meters.

In the operational processing system, incoming Slocum observations were received in real-time, processed into binary files by the Slocum Dock Server software, and then processed further and written to TESAC KKYY-formatted files by software installed by John Kerfoot from RUCOOL. Subsequently, the KKYY files were converted into ASCII M2K-formatted files. The M2K format is one of the standard formats used at NAVO to import and export ocean temperature and salinity profiles to and from the NAVO MOODS global profile archive. The KKYY files store the temperature and salinity to a precision of one hundredth degree Celsius and one hundredth PSU, respectively, and the depth is stored to the nearest meter. At the same time, observations received from the Seagliders were first formatted into Netcdf files by the Base Station system and then converted to M2K files. The final set of files for both Slocums and Seagliders stored observations as separate ascending or descending profiles extracted from each glider dive. The depths were stored at the nearest whole 1-m value and the temperature and salinity were stored at 0.01 precision. The final step in the operational processing scheme

was to load the M2K files into the NOPEDS GUI-based manual editor and manually view and edit each profile. The NOPEDS editor allows the user to delete or modify individual observations or to flag the entire temperature or salinity profile as bad.

At the same time, the LAGER system obtained the raw Seaglider Netcdf data files from the Base Station system and the raw binary Slocum data files from the Dock Server system. The raw data files were then processed through the automated QC software of LAGER. Some files were flagged for further evaluation in the LAGER manual QC GUI. However, the purpose of this report is to validate the results from the automated QC performed by LAGER. Therefore, for purposes of this study, the subsequent manual evaluation and editing that would normally be performed in the LAGER manual QC GUI was eliminated.

### **3 Discussion of typical CTD errors associated with gliders**

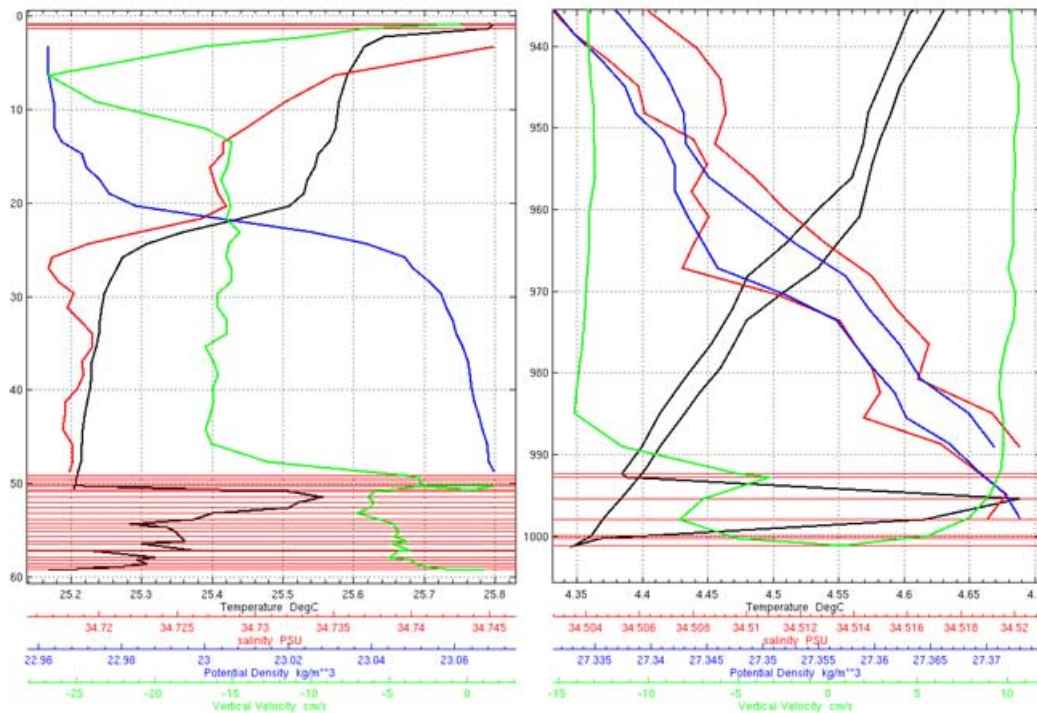
The algorithms usually applied to the task of detecting errors in CTD or XBT profile observations often fail to detect some of the most prevalent glider CTD errors. This section presents a short discussion of a few of the glider-specific CTD errors which the LAGER QC algorithms must identify.

#### **3.1 Seagliders**

For both Seagliders and Slocums the source of many CTD errors can be traced to low speed of the glider. Since the speed of the glider through the water is not known accurately, the LAGER software substitutes vertical speed, computed from the change in pressure versus time, for total glider speed when performing the QC analysis. Typically, the descent and ascent vertical velocity of the Seaglider remains between 10 cm/s and 15 cm/s. When the vertical velocity slows to less than about 5 cm/s or 6 cm/s, the speed of flow through the CTD conductivity cell is apparently too low to provide a measurement of conductivity consistent with temperature, resulting in an inaccurate calculation of salinity. The LAGER v1.0 Manual discusses the procedure used to correct the conductivity measurements for errors caused by thermal inertia of the conductivity cell. At the lowest speeds, the correction often appears to fail, resulting in obviously erroneous values of the computed salinity. The correction algorithm assumes a constant (medium) flow rate through the conductivity cell, and the flow rate during periods when the glider's vertical velocity is below 5 cm/s is well below that medium rate. However during certain situations, the CTD errors related to low speed are apparently due to more than just the thermal inertia of the conductivity cell. One of these situations is shown in Figure 3.

Both frames in Figure 3 show profiles of temperature, salinity, potential density, and vertical velocity versus depth measured by sg137 during the RIMPAC 2008 exercise. The frame on the left shows only the descending portion of a short profile that ends at a depth of about 60 m and the frame on the right shows both the descending and ascending profiles in the lowest 60 m of a deep 1000-m profile. The profile of vertical velocity

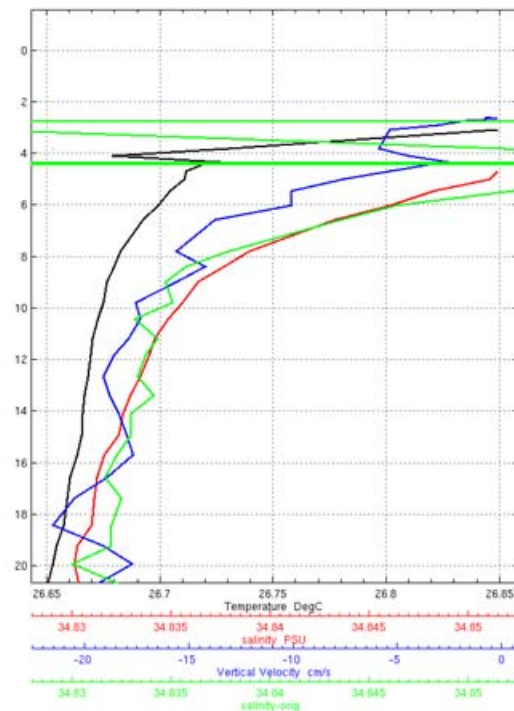
(negative on the descent) indicates that the glider rapidly deaccelerates starting at about 15 m from the bottom of the profile as it prepares to turn around and begin the ascent. In both cases the acceleration appears to overshoot momentarily and stalls out on the shorter profile about 10 m from the bottom. Many shorter profiles show the glider actually reversing (vertical) direction momentarily, performing a short loop or skip. The surprising effect of this sudden stop is that the temperature suddenly increases by  $0.35^{\circ}\text{C}$  in both cases to about the same temperature as measured higher up on the profile, 30 m above this depth with the shorter profile and 60 m above this depth on the deeper profile. Apparently, water trapped in the hull of the glider is suddenly released and passes over the CTD thermistor. The horizontal red lines in Figure 3 indicate the depths where LAGER flagged the temperature observations as being bad. The Seaglider design was optimized for deep profiles down to 1000 m, and the sudden temperature increase is often found near the bottom of shorter profiles, but not usually on the deeper profiles. LAGER almost always detects and removes errors caused by this peculiar Seaglider idiosyncrasy; LAGER missed only one bad point of one profile near the bottom during the RIMPAC 2008 exercise.



**Figure 3** Profiles of temperature (black), salinity (red), potential density (blue), and vertical velocity (green) measured by Seaglider sg137. The bottom of the profile on the left is near 60 m and the profile on the right ends near 1000 m depth. In both cases, the vertical velocity of the glider suddenly deaccelerates beginning at about 15 m from the bottom, resulting in a sudden erroneous temperature increase. The horizontal red lines mark the depths where LAGER flagged the observations as bad. The frame on the left shows only the profile measured on the descent and the frame on the right shows the descending and ascending profiles. The sudden temperature increase occurs only on the descending profile.

A similar problem occurs sometimes near the surface of descending profiles measured by the Seaglider. Figure 4 shows the initial 20 m of a descending profile. Once the vertical

velocity reaches about 5 cm/s, the temperature suddenly decreases by about 0.04° C and then quickly recovers. At and above the depth of the spike, the uncorrected salinity fluctuates wildly. In Figure 4, the green line is the salinity computed from the measured temperature, conductivity, and pressure. The red line is the salinity computed from the measured temperature and pressure, but the conductivity has been corrected for the effects of the thermal inertia of the conductivity cell (and the points flagged by LAGER have been removed from the corrected conductivity). LAGER usually detects these errors and flags them. In the case shown in Figure 4, all CTD measurements from the base of the temperature spike to the surface were flagged as bad.



**Figure 4** Profiles of temperature (black), uncorrected salinity (green), corrected salinity (red), and vertical velocity measured from Seaglider sg137 near the surface during the initial descent. The sudden erroneous 0.04° C spike in temperature at 4 m depth occurs as the vertical velocity reaches about 5 cm/s. The uncorrected salinity varies wildly (going off the plot) in the block of depths from the base of the spike to the surface.

### 3.2 Slocums

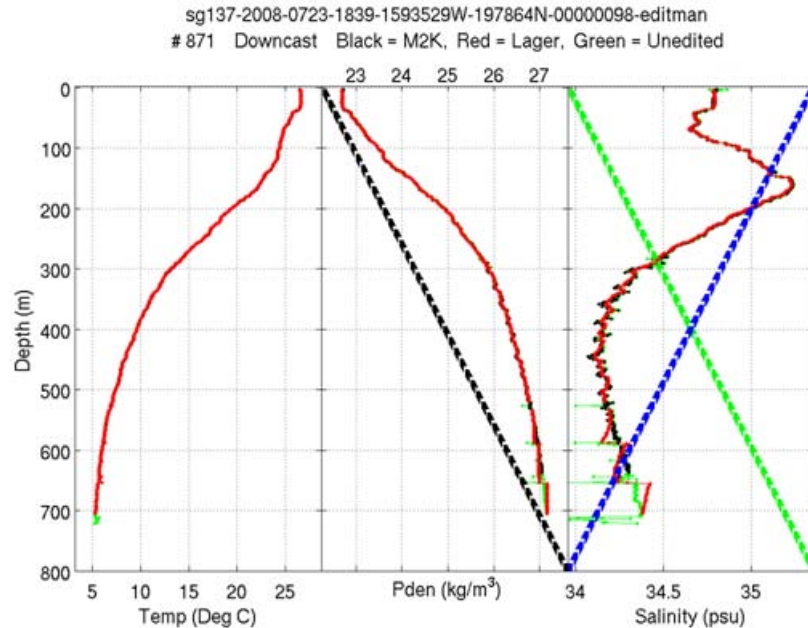
As with the Seaglider, CTD measurements made with the Slocum, particularly the conductivity measurements, become questionable when the vertical velocity drops below about 5 cm/s. However, unlike the Seaglider, the Slocum doesn't exhibit the sequence of sudden deacceleration, stalling, and recovery near the bottom. Nor does the temperature appear to exhibit the temperature spikes associated with sudden change in velocity at low speeds. However, the Slocum is prone to another type of problem that resulted in the loss of many observations during RIMPAC 2008. It is a issue that can be overcome with careful planning and with adequate prior knowledge of the characteristics and causes of the problem. Observations made by the CTD and various optics instruments are

transferred from the science bay and combined with navigation and various system data as the glider is flying. The observational data are accumulated in a file that is later transmitted to the Iridium Satellite communication system. The transfer of data from the science bay is made over a buffer that can accumulate data until the main computer has time to accept them. Transfer delays can occur near the start of a dive while the computer is busy with flight calculations. Later, when time permits, the delayed data is combined with the other system data and stored. The data file received at NAVO has one line of data at each time interval (using the main system clock). When the data from the science bay are delayed at a particular time, the science data observed at that time is set to missing values in that line in the file. However, at a later time, once the science data is extracted from the buffer, it is added to the main file of data at the present system clock time. To properly interpret the delayed science bay data, the time from one of the science bay instruments, usually the CTD clock time, and the pressure measured by the CTD must be sent at the same time as the temperature, salinity, and optics observations. Depending on the amount of science data being sent back at each glider surfacing, the time delay between the system clock and the science clock can reach a maximum in the range from 70 seconds to about 200 seconds. If too much data is being sent from the science bay, and the delay is too long, the buffer will overflow and the data is lost. The RIMPAC 2008 exercise was the first major use of Slocums by the NAVO GOC and the first real-time use of Slocum data in LAGER. As a result, the data-delay and buffer-overflow problems were not fully understood at the beginning of the exercise, and many changes to the glider control and to the LAGER software were made throughout the exercise to compensate for these problems.

## 4 Validation

The results from the LAGER automated QC system were evaluated against the results from the NAVO operational processing system by visually viewing and comparing plots of temperature, salinity and potential density versus depth prepared from each glider profile. Figure 5 shows an example of one of these comparison plots. The plot consists of three side-by-side frames. The frames, from left to right, display temperature versus depth, potential density versus depth, and salinity versus depth. Three profiles are drawn in each frame. The first profile drawn is a green line of the original unedited profile. The second profile drawn is the black line (labeled M2K) of the final profile from the operational NAVO system. The third line, drawn in red, is the result from the LAGER automated QC. All observations flagged as bad by LAGER were eliminated before the red line (of temperature, salinity, or potential density) was drawn. Therefore, if LAGER correctly detected and flagged all bad observations, then the red line should be error free. If the values from all three profiles are equal, then only the final red line (the last drawn) will show. To allow rapid visual recognition, thick diagonal dashed lines are drawn across the plots to indicate that a particular flag has been set for the results from one of the two systems. The black dashed line is displayed only if the NAVO system (NOPEDS) determined that the entire temperature or salinity profile should be deleted. The green dashed line is drawn only if LAGER indicated that this profile should be sent to the manual editor for further review. The thick green dashed line should only be drawn if there are errors remaining on the thin red line of the LAGER-processed values.

Otherwise, the thick green dashed line is considered a false alarm (file sent for manual QC when it wasn't required). The blue dashed line is drawn over the temperature or salinity plot only if the LAGER automated QC determined that that temperature or salinity profiles should be deleted.



**Figure 5** Example of the evaluation plots used to compare profiles processed by the NAVO QC system against those processed by the LAGER automated QC system. In each frame, three (thin) lines are drawn. The green line shows the raw unprocessed values, the black line shows the results from the NAVO QC system, and the red line shows the results from the LAGER automated QC system. The thick black dashed line is present only if the NAVO system determined that the salinity or temperature profile should be deleted. The thick green line is present only if LAGER determined that the profile should be sent for further evaluation and editing in the manual QC GUI. The thick blue dashed line is present in the temperature or salinity plot only if that entire profile was flagged as bad by LAGER.

To compile the statistics for the validation, the comparison plot for each set of profiles was viewed and two questions were answered for each. The goal of viewing these plots was to compile the number of errors of two types. The first type of error was determined to have occurred if the LAGER-processed profile (thin red line) that had at least one undetected remaining error and the profile had not been flagged for further evaluation by the LAGER manual QC. The second type of error was determined to have occurred if there were no remaining errors on the red line, but the file had been flagged for further manual QC by LAGER. These and other statistics are listed in Table 1. This table has a row for each ascending and each descending set of profiles from each glider. The columns are the glider name, whether the profile was descending (Dn) or ascending (Up), the total number of profiles evaluated, the number of profiles sent for further manual evaluation by LAGER, the number of false alarms (the profile was flagged for further manual QC by LAGER, but there were no visible errors remaining on the profile), the number of profiles that were not sent to manual QC, but should have been (undetected errors remain on the LAGER profile), the number of LAGER temperature profiles that

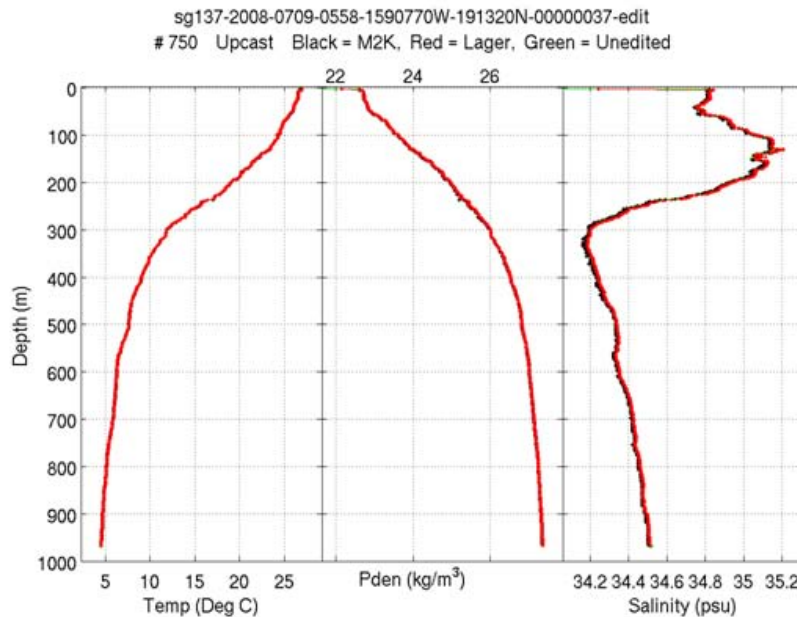
were marked for deletion, and the number of LAGER salinity profiles that were marked for deletion by the automated LAGER QC software.

**Table 1** Evaluation statistics of the LAGER automated quality control analysis. For each glider and profile direction (ascending /Up and descending/Dn), this table lists the total number of profiles evaluated, the number of profile flagged for further evaluation in the manual QC GUI, the number of profile sent to manual QC that should not have been sent (false alarms), the number of profiles not sent to manual QC that should have been, and the number of full temperature profiles and the number of full salinity profiles identified for deletion by the LAGER automated QC analysis.

<b>Glider Name</b>	<b>Dir</b>	<b>Total num</b>	<b>Num to manual</b>	<b>Num false alarms to manual</b>	<b>Num bad not sent to manual</b>	<b>Num bad Temp profs</b>	<b>Num bad Salt profs</b>
sg128	Up	202	1	1	0	0	0
sg128	Dn	204	1	1	1	0	0
sg136	Up	136	0	0	0	0	0
sg136	Dn	136	0	0	0	0	0
sg137	Up	118	3	3	12	0	0
sg137	Dn	115	11	8	0	0	8
sg138	Up	152	71	0	0	0	65
sg138	Dn	152	1	0	0	0	0
sl005	Up	0	0	0	0	0	0
sl005	Dn	186	0	0	0	0	0
sl016	Up	0	0	0	0	0	0
sl016	Dn	832	45	16	0	0	13
sl113	Up	1	0	0	0	0	0
sl113	Dn	893	3	0	0	0	0
sl114	Up	0	0	0	0	0	0
sl114	dn	530	165	5	0	1	145

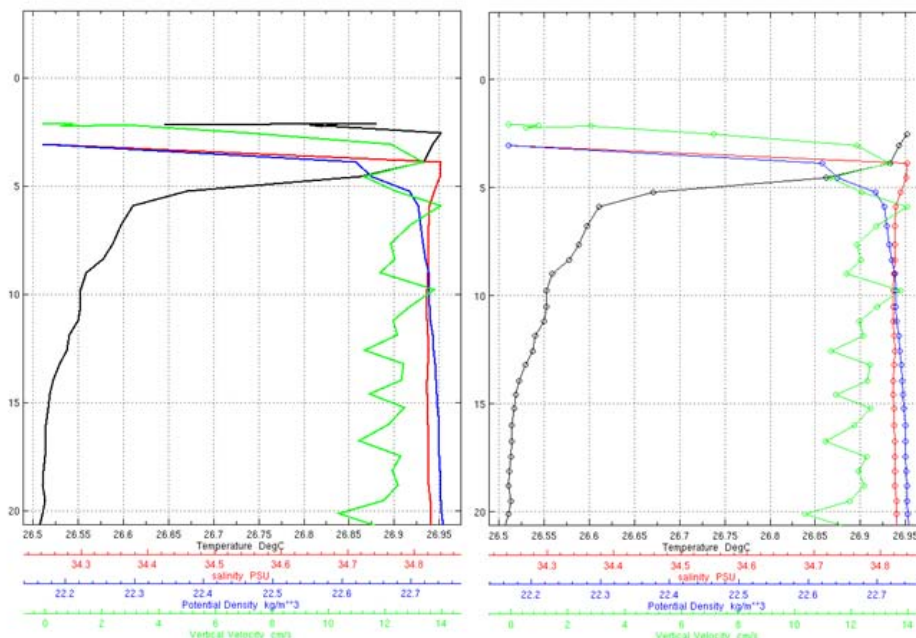
The total number of profiles evaluated in Table 1 is 3657. A total of 301 profiles (8.2%) were sent for further evaluation in the LAGER manual QC step. Of these, 34 (11% of the 301 profiles and less than 1% of the total number of profiles) were false alarms. Only 13 LAGER-processed profiles (0.36 % of the total) with at least one remaining error were not sent for further manual evaluation. All but one of these errors occurred on the ascending profiles of Seaglider sg137. In each of these cases, the error was due to a single salinity at the top remaining point of the profile that was from 0.2 psu to 0.6 psu lower than the salinity at the next deeper depth.

Figure 6 shows the evaluation plot for the most serious of the cases (dive 27 from Seaglider 137) where an error was not detected by LAGER. The sudden shift in salinity by about 0.6 psu is shown more clearly in the plot (extracted from two screen shots of the Manual QC GUI) of the upper 20 m of the profile in Figure 7. This figure shows that the sudden decrease in salinity occurred while the vertical velocity of the glider was still about 14 cm/s and while temperature was apparently good. The QC algorithms in the next version of LAGER can easily be modified to detect and flag this type of error.

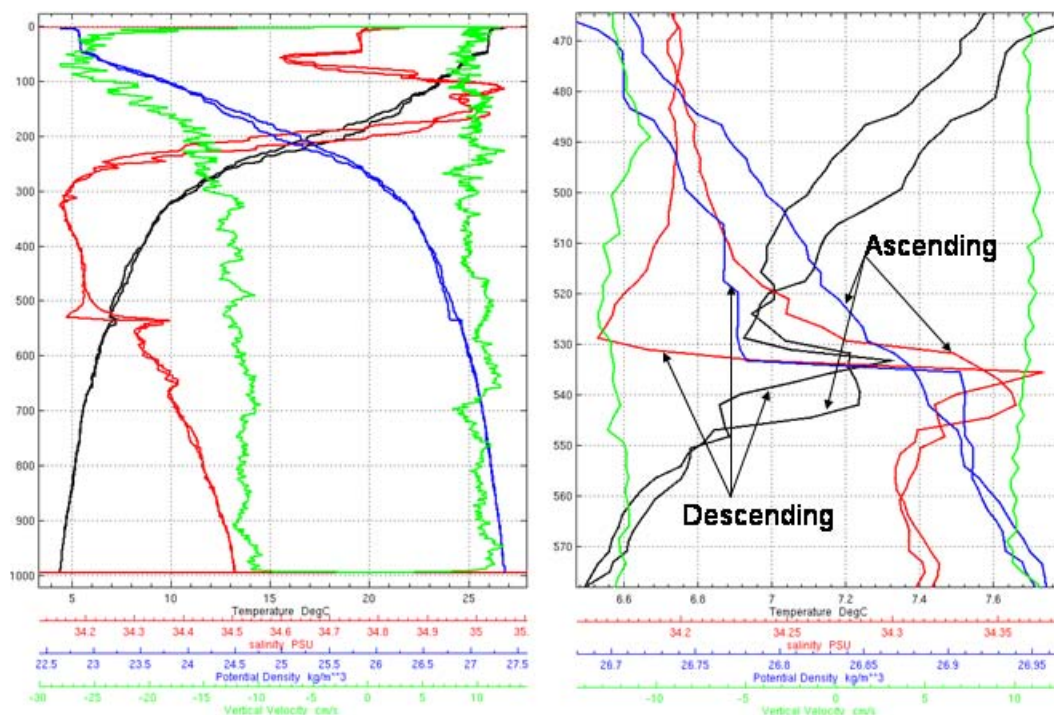


**Figure 6** Evaluation plot for ascending profile from dive 37 of Seaglider sg137 showing a probable salinity error near the surface that was not detected by the LAGER automated QC software. The faint green line across the top of the salinity plot shows the original raw salinity profile (the remaining portion of the original profile is covered by the red and black profiles lines).

The only other case where an error was left undetected by LAGER and also the file was not flagged to be forwarded for further manual QC was on the descending profile of dive 37 of Seaglider sg137. Figure 8 shows plots of the descending and ascending profiles from this dive. LAGER detected a static instability in the profile near a depth of 135 m and applied the stability correction algorithm. The final profile is stable as can be seen by the profile of potential density, but the resulting salinity profile (modified by the algorithm) looks unrealistic and doesn't match the form of the salinity in the ascending profile. The instability occurs at the location of an unusual thermal inversion, seen on both the ascending and descending profiles. The stability correction algorithm modifies only the salinity profile to obtain a stable profile. The stability correction algorithms usually produce a realistic modification of the salinity profile to achieve stability, but failed in this case. The stability algorithm will be upgraded in the next version of LAGER to observe constraints on the modifications similar to those employed in the algorithm discussed in Jackett and McDougall (1995).

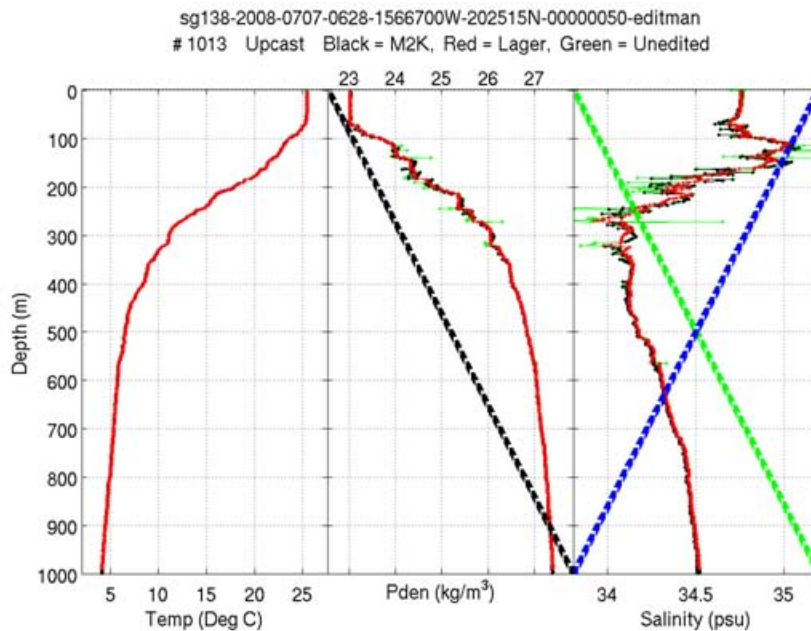


**Figure 7** Plots of the temperature, salinity, potential density and vertical velocity profiles from the upper 20 m of the ascent from dive 37 of Seaglider 137. The left frame shows all profiles values, including those flagged as bad by the LAGER automated QC. The frame on the right shows only points that were not flagged as bad.

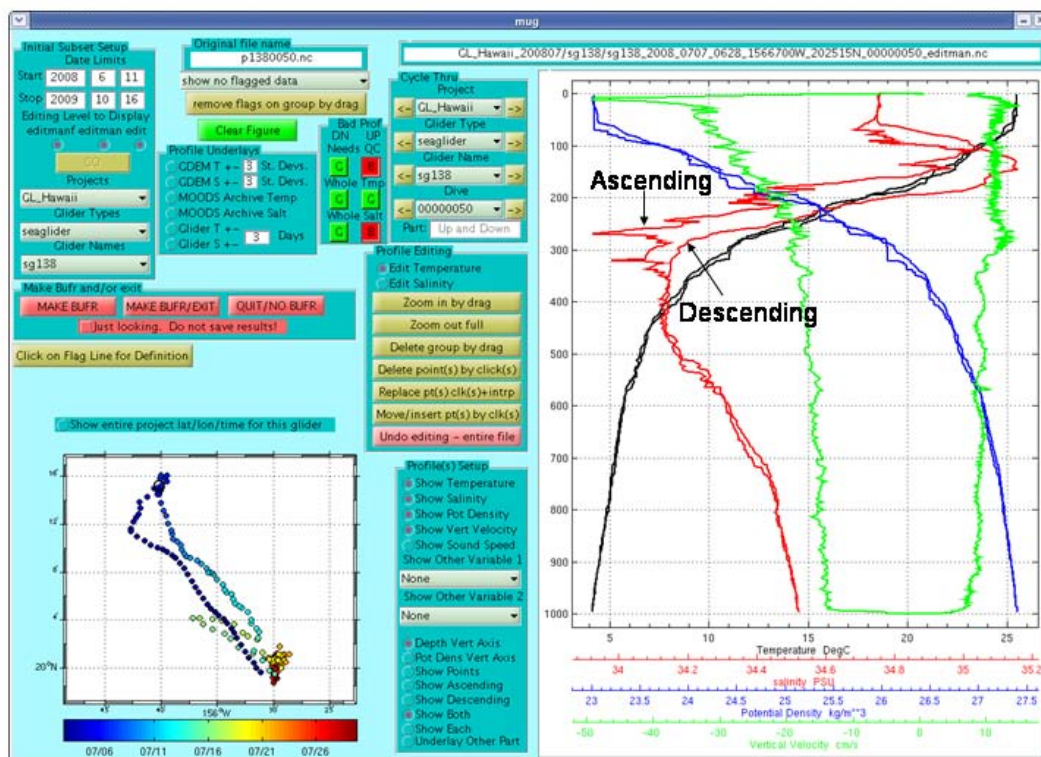


**Figure 8** Plots of the ascending and descending temperature, salinity, potential density, and vertical velocity from dive 27 of Seaglider 137. The left frame shows the entire profile, and the right frame shows an enlargement of the depth range near the middle of the profile where the stability correction applied to the descending profile by the LAGER automated QC software was not successful.

Three of the gliders had large numbers of profiles flagged for further review by the LAGER manual QC: 71 (47%) of the ascending profiles of sg138, 45 (5%) of the descending profiles from sl016, and 165 (31%) of the descending profiles from sl114. Seaglider sg138 developed a problem where conductivity measurements were erratic on the ascent, but not the descent. An example of one of these profiles is shown in Figure 9. The LAGER automated QC software checks for spikes in the measured conductivity, but the spikes are often too small to trigger a flag. Instead, LAGER first corrected the conductivity for the thermal inertia lag and computed salinity. LAGER then detected a large number of instabilities due to the erratic computed salinity (caused by the erratic conductivity). Then, LAGER stabilized the profile by making local changes to salinity around the instabilities. The final salinity profile (red line in Figure 9) is smoother than the raw salinity, but LAGER still sends it to the manual QC step because of the large number instabilities found before stabilization. When the profile reaches the manual QC step, the operator, when viewing the GUI screen shown in Figure 10, would find that the ascending profile is offset from the descending profile and much noisier than the descending profile and therefore would not change the red stoplight that indicates that the full salinity profile is to be deleted (as set by the automated QC software).

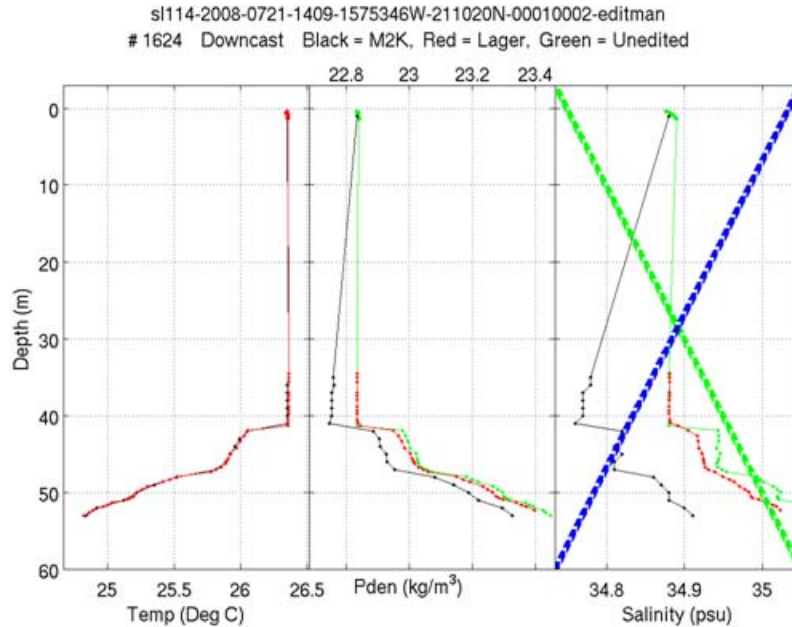


**Figure 9** Example of one of the large number of ascending profiles from Seaglider sg137 exhibiting erratic conductivity measurements. This profile was flagged for deletion by LAGER and flagged for further review by manual QC. The thick black dashed line also indicates that it was flagged for deletion by the NAVO NOPEDS QC.



**Figure 10** Screen shot of the MUG GUI showing profiles from Seaglider sg138 dive 50. The ascending full salinity profile was flagged bad as indicated by the red stop lights shown in the upper left center of the screen.

A large number of descending profiles from Slocum sl114 were flagged for further review by the LAGER manual QC due to the large data gaps detected in the profiles. The gaps were the result of the buffer overflow that occurred when too much data tried to pass from the science bay to the system computer through the buffer (discussed in section 3.2). Figure 11 shows an example of a descending profile from sl114 with a large gap in observations from about 2 m to 35 m depth. LAGER flagged this profile for deletion and sent it for further review by the manual QC. The linear trend in the salinity profile processed by the NAVO processing system is due to an error that was later corrected.



**Figure 11** Descending profile from Slocum sl114 showing a large data gap from about 2 m to 35 m depth caused by a buffer overflow. The linear trend in the M2K (NAVO processing scheme) was due to an error that was later corrected.

## 5 Conclusions

The main goals of the LAGER automated QC analysis are to detect and flag all observational errors. Since LAGER cannot detect and flag all errors, it was designed to send a file to the manual QC step for further review if it determines that there might be remaining unflagged errors. However, LAGER was tuned to limit the number of profiles that it sends to manual QC while, at the same time, insuring that very few undetected errors slip through the combined automated and manual QC system. The validation was performed by comparing the results from the LAGER automated QC with the results from the operational QC system employed by NAVO. The validation results summarized in Table 1 show that 8.2% of the 3657 profiles used in this validation study were sent to the manual QC step for further evaluation. This percentage is larger than expected, but is the result of two deficiencies in the gliders, one caused by a hardware failure in a Seaglider and the other due to large data gaps caused by a recurring buffer overflow problem in some of the Slocums. The number of profiles with errors that were undetected in the manual QC and that were not sent for further evaluation in the manual QC was low (13 or 0.36% of the total). Of these errors, most (12) were due to a single salinity error at the top of each profile, and the other (1) was due to an inadequate performance of the profile stabilization algorithm. We expect that both types of errors can be either detected or removed with software upgrades in the next version of LAGER.

## 6 References

Carnes, M., 2008, LAGER Manual (Local Automated Glider Editing Routine Version 1.0, Unpublished Manuscript, Naval Research Laboratory, Stennis Space Center, MS, 51 pp.

Jackett, D.R. and T.J. McDougall, 1995, Minimal adjustment of hydrographic profiles to achieve static stability, *J. Atmos. and Ocean. Tech.*, 12, 381-389.

Marks, K.M. and W.H.F. Smith, 2006, An evaluation of publicly available bathymetry grids, *Mar. Geophys. Researches*, 27, 19-34.

## **7 APPENDIX A**

### **LAGER Manual (Local Automated Glider Editing Routine) Version 1.0**

Michael R. Carnes  
Naval Research Laboratory  
Code 7332  
July 29, 2008

# TABLE OF CONTENTS

1	INTRODUCTION .....	20
2	INSTALLATION .....	21
2.1	LAGER software .....	21
2.2	Set home path.....	21
2.3	m_map.....	21
2.4	SEAWATER.....	21
2.5	MATLAB NetCDF toolbox .....	21
2.6	Install BUFR.....	22
2.7	Compile Bufr .....	22
2.8	Newnames.....	22
2.9	Areas .....	23
2.10	glider_qc_info.m.....	24
3	CONFIGURATION.....	24
3.1	Flow diagram .....	24
3.2	Data base.....	26
3.3	File names .....	26
3.3.1	Seaglider initial .....	26
3.3.2	Slocum initial .....	27
3.3.3	LAGER NetCDF after import.....	27
3.3.4	BUFR file.....	28
4	Import Glider ObservationS.....	28
4.1	Seaglider .....	28
4.2	Slocum .....	30
5	Automated QC .....	33
5.1	Quality Control tests and flag setting at each depth .....	33
5.1.1	Depth check T and S .....	33
5.1.2	Global bounds check T .....	33
5.1.3	Comparison to GDEM T.....	33
5.1.4	Spike test T .....	34
5.1.5	Gradient test T.....	34
5.1.6	Spike test C .....	34
5.1.7	Vertical velocity test .....	34
5.1.7.1	Slocum .....	34
5.1.7.2	Seaglider .....	34
5.1.8	Surface Chop.....	35
5.1.8.1	Slocum .....	35
5.1.8.2	Seaglider .....	35
5.1.9	Global bounds check S.....	35
5.1.10	Comparison to GDEM, S .....	35
5.1.11	Spike test S.....	35
5.1.12	Gradient test S.....	35
5.2	Manual Flags.....	36
5.2.1	temperature .....	36
5.2.2	salinity.....	36
5.3	Overall quality flags.....	36

5.3.1	keeptemp_flag.....	36
5.3.2	keepsalt_flag .....	36
5.3.3	needs_manual_editing_flag .....	36
5.4	Conductivity correction .....	37
5.5	Static stability.....	40
6	BUFR File Generation .....	42
7	MUG (manual Utility for gliders).....	46
7.1	General operation.....	46
7.2	Initial setup.....	48
7.3	Cycling through profiles .....	50
7.4	Profile display options .....	52
7.5	Underlay comparisons .....	54
7.6	Editing temperature and salinity profiles.....	54
7.7	Changing and displaying flags.....	57
7.8	Exit, Quit, and generating BUFR file .....	58
8	APPENDIX A .....	59
9	APPENDIX B .....	61
10	APPENDIX C .....	62
11	APPENDIX D .....	65
12	REFERENCES .....	67

# 1 INTRODUCTION

The purpose of this manual is to document the LAGER (Local Automated Glider Editing Routine) ocean glider quality control system that is being developed for use by the Naval Oceanographic Office (NOO). The NOO currently operates several gliders, including Seagliders (Eriksen et al., 2001) developed in a collaborative effort between the Applied Physics Laboratory University of Washington and the University of Washington School of Oceanography and Slocum Shallow Battery gliders developed by Webb Research. NOO will soon begin using Spray gliders (Sherman et al., 2001) developed under ONR support by Scripps and Woods Hole scientists. Ocean gliders are autonomous platforms which fly in a saw-tooth-sampling pattern in the upper ocean by changing their buoyancy. Depending upon configuration, gliders sample profiles of pressure, temperature, and conductivity as well as various ocean optical parameters. The gliders surface at regular intervals to transmit their observations over the Iridium Satellite LLC data communications system consisting of 66 low-earth orbiting satellites. The Seaglider make a complete data transmission after every dive, but the Slocum may complete several dives before surfacing and transmitting data.

The data from the gliders is received at NOO in real time where it is reformatted and undergoes manual quality control (QC) before it is sent on to various applications. Gliders provide the Navy with the capability to covertly insert assets over the horizon into denied areas to gather ocean observations used in antisubmarine warfare and mine counter measures applications. To be most effective, the incoming glider data must be processed rapidly and routed to the various applications. However, the requirement for manual QC of the observations and other procedures presently being applied often delays use of these data by one day. The LAGER system is being implemented to reduce the time and manpower required to process and QC glider observations. The basic LAGER processing scheme is described in the next paragraph.

Within minutes of receipt of data from the Iridium system, LAGER reformats the glider data of all types into a single standard file format and then processes the data through a series of automatic QC tests, corrects conductivity and salinity, and applies calibrations to some types of optics observations. Both the raw observation files in the original file format and the standard-format files that have passed through QC are placed in the glider data flat file archive. The automatic QC software also determines whether the profile needs further manual QC. Each file identified as needing manual QC, is marked to indicate that requirement before being stored in the glider archive. If not required, the glider data is reformatted as single profiles into BUFR-format files (a international standard table-driven format used to transfer data between operational centers) and sent to the NOO Real Time Data Handling System (RTDHS). From there, the data is passed to various applications. As time permits, glider pilots in the NOO Glider Operations Center (GLOC) edit files marked for manual QC using the LAGER GUI-based editor (named MUG). Once manually edited, the files are reformatted into BUFR and sent to the RTDHS.

Each of the steps in the LAGER processing scheme as well as the infrastructure that supports it is described in detail in the following sections.

## **2 INSTALLATION**

The steps to installing the LAGER system are listed below:

### **2.1 LAGER software**

Install the LAGER software within the subdirectories of the main directory, LAGER. The software must be installed on a Linux computer system. Presently, all software is written in MATLAB, Fortran (g77), and csh. The LAGER software directory tree and the software within each directory are listed in Appendix A.

### **2.2 Set home path.**

Set the path environment variable, `glider_qc_home`, equal to the full path to the LAGER main directory. In the csh shell, this is accomplished by adding a line to the `.cshrc` file in the user's home directory such as, for example:

```
setenv glider_qc_home /full_path_to_LAGER/LAGER.
```

If you are running in the bash shell, then insert the following line (for example) in the `.profile` file in your home directory:

```
export glider_qc_home = /full_path_to_LAGER/LAGER.
```

### **2.3 m\_map**

Install the `m_map` directories and all MATLAB files and subdirectories contained within them on a disc accessible to the LAGER software. The files within the `m_map` directory are derived from the toolkit available at <http://www.eos.ubc.ca/~rich/map.html>.

However, the LAGER installation contains an extra file, `mygrid_dbdbv2.m`, in the `m_map` directory and an extra NetCDF data file, `dbdb2_v2b.nc` in the `m_map/private` subdirectory.

### **2.4 SEAWATER**

Install the SEAWATER directory and all files contained within it on a disc accessible to the LAGER software. The files in SEAWATER toolkit were obtained from the developer's web site at [http://www.cmar.csiro.au/datacentre/ext\\_docs/seawater.htm](http://www.cmar.csiro.au/datacentre/ext_docs/seawater.htm).

However, in the LAGER installation, the `sw_svel.m` has been updated to compute sound speed using the new sound speed equations and modifications found in Millero and Li (1994) and Chen and Millero (1977), which is the Navy standard algorithm in the Oceanographic and Atmospheric Master Library (OAML).

### **2.5 MATLAB NetCDF toolbox**

Install the MATLAB NetCDF toolbox. Download MEXNC from <http://mexcdf.sourceforge.net> and install as directed by its installation documentation. Next, download and install the NetCDF Toolbox from [http://mexcdf.sourceforge.net/netcdf\\_toolbox/netcdf\\_toolbox.html](http://mexcdf.sourceforge.net/netcdf_toolbox/netcdf_toolbox.html). After installation, test it by running `tnetcdf` from MATLAB.

## 2.6 Install BUFR

Install ECMWF BUFR library and tables. Make a directory to hold the final BUFR library and tables (change directory to the location where a new directory is to be placed. Type in `mkdir BUFRLIB`). Download the `bufr_000351.tar.gz` from <http://www.ecmwf.int/products/data/software/download/bufr.html>. Un-gzip (`gunzip bufr_000351.tar.gz`) and untar (`tar -xvf bufr_000351.tar`) this file to create the directory, `bufr_000350`. Change directory into `bufr_000350` and edit line 129 of the file named `build_library`. Change `_gFortran` to `_gnu`. Save and exit `build_library`. Change directory to the `config` subdirectory. Edit `config.linux_gnu`. On line 17, which should start with `FFLAGS -`, remove the section `"-fno-second-underscore"`. This might not be necessary on some Linux installations and machine types. Save and exit and change directory back to the `bufr_000351` directory. Type in, `./build_library`. Answer the three questions asked by this program as: (1) use `gnu`, (2) keep 32 bit, (3) put results in (the full path to the `BUFRLIB` directory created earlier). Next, (while still in directory `bufr_000351`) type in, `./install`. When finished change directory to `bufr_000351/examples`. Edit `Makefile`. On line 21 add `create_bufr` and remove the last 2 entries (result should look like `"EXECs - decode bufr bufr_decode create_bufr tdexp tdexp"`). Save and exit. Type `"make"`. When compilation and linking is complete, run the program, `create_bufr` (type in `create_bufr`). If all goes well, a large amount of output will be printed to the screen. Finally, check in the `BUFRLIB` directory. It should contain a file named `libbufr.a` and a subdirectory named `bufrtables`.

## 2.7 Compile Bufr

Compile the Bufr Fortran programs in LAGER. Change directory to `$glider_qc_home/Bufr`. Edit `Makefile`. On lines 14, 50 and 53 change the path name of the BUFR library to the full path of the `BUFRLIB` directory made earlier. Save and exit. Type `"make"` to compile and link `decode_bufr` and `glider_nc_to_bufr`.

## 2.8 Newnames

Edit the `Newnames` file in `$glider_qc_home/INFOFILES`. An example of the `Newnames` file is shown next.

```
#
# old name          new name
#
ru05                sl005
ru16                sl016
unit_114            sl114
harpo77             sl077
chico79             sl079
groucho82           sl082
zeppo83             sl083
```

Lines beginning with `#` are comments. More comments line can be added if desired. The following lines show a pair of glider names on each line. There must be at least one space on the line between the two names. The name on the left hand side is the true name of the glider. It is the name used for this glider on standard files (such as the `sbd` files described later) generated by Webb Research software from observations made by this

glider. The name on the right is the name of this glider in the LAGER system. The glider names used in LAGER are always five characters long, lower case, and have the form, sANNN, where A is a g for seaglidern, an l for slocums, and a p for sprays. The NNN represent a three digit number which is zero-filled at the front. When an existing glider name, such as those used by Rutgers University and by NRL does not conform to the LAGER convention, it is converted using information from the Newnames table when it is imported into the LAGER system.

## 2.9 Areas

Edit the Areas file in \$glider\_qc\_home/INFOFILES. The destination directory in the final LAGER archive of each glider file is determined by information contained in the Areas file. An example of an Areas file is shown next:

```
#EXERCISE NAMES|S|N|E|W|START TIME|END TIME|gliders
#
BIOSPACE_200804|999|999|999|999|20080401|20080331|s1005,s1016,s1077,s1079,s1082,s1083
BIOSPACE_200805|999|999|999|999|20080501|20080531|s1005,s1016,s1077,s1079,s1082,s1083
BIOSPACE_200806|999|999|999|999|20080601|20080630|s1005,s1016,s1077,s1079,s1082,s1083
#
GL_Hawaii_200806|999|999|999|999|20080601|20080630|sg128,sg136,sg137,sg138
GL_Hawaii_200807|999|999|999|999|20080701|20080731|sg128,sg136,sg137,sg138,s1016,s1114,
s1113,s1005
GL_Hawaii_200808|999|999|999|999|20080801|20080831|sg128,sg136,sg137,sg138
```

Lines beginning with # are comments. More comments line can be added if desired. Each line begins with an area or project name with an appended year and month (e.g., BIOSPACE\_200806). This name is the project/area subdirectory name of the LAGER glider data base archive (see the Data Base Section below). Blank spaces are allowed between individual elements even though none are shown in the example above. Long lines cannot be continued on the next line after a return (or control-linefeed or newline); the line above starting with GL\_Hawaii\_200807 appears to be continued on a second line, but that is due to text formatting in this document. In the Areas file, these two lines are contained on a single line. Next, the geographic boundaries are defined by the southern boundary latitude, the northern boundary latitude, the eastern boundary longitude and the western boundary longitude, all defined in decimal degrees and separated by vertical bars (|). If the geographic boundaries are to be left undefined on this line, then replace each value by 999. Following the geographic boundaries are the beginning and ending dates for this project/area. The beginning and ending dates (normally the first and last days of a month) are in the form, YYYYMMDD, where YYYY is the four-digit year, MM is the two-digit month (filled with zero on the front if less than 10), and DD is the two-digit day (filled with zero on the front if less than 10). After the dates, the name of each glider flown in this project/area contained with the geographic area and within the date limits is listed, separated by commas. The glider names listed in the Areas file must use the LAGER glider name convention. If the true name of the glider doesn't match the LAGER convention, then the alternative name defined in the Newnames file, discussed above, must be used. If, during LAGER processing, a glider is found not to belong to any of the projects/areas defined in the Areas file, it is placed in the orphans subdirectory of the LAGER data base archive.

## 2.10 *glider\_qc\_info.m*

Edit the `glider_qc_info.m` file in `$glider_qc_home/INFOFILES`. An example listing of the `glider_qc_info.m` file is shown in Appendix B. This file defines the paths to many of the directories used by LAGER. The first two directories, `setup.incoming_seaglider_dir` and `setup.incoming_slocum_dir`, are the directories where LAGER first reads Seaglider data NetCDF files coming from the Basestation and where it first reads Slocum sbd files coming from Docserver, respectively. The `setup.going_to_rtdhs_dir` is the path to the directory where edited profiles in BUFR format are sent by LAGER. Files in this directory will be accessed by the RTDHS (Real Time Data Handling System) which routes them to various applications. The base directory of the glider data hierarchical archive filing system is defined by `setup.base_qc_data_directory`. Make this directory if it does not already exist. LAGER puts raw, unedited files as well as various versions of edited glider files into subdirectories of this directory.

## 3 CONFIGURATION

### 3.1 *Flow diagram*

The flow of glider observations through the LAGER system is illustrated in Figure 1. The upper layer displays the types of incoming and outgoing files. Data transmitted over the Iridium Satellite system from Seagliders are received by the Basestation system and processed into the standard Seaglider NetCDF files. Slocums over the Iridium system are received by the DockServer system and written to sbd (short binary data) files. After being processed through the LAGER system, the LAGER NetCDF files are converted to BUFR files, each containing a single ascending or descending profile of temperature, salinity, depth, time, latitude and longitude. The BUFR-formatted files are then picked up by the Naval Oceanographic Office Real Time Data Handling System (RTDHS) for distribution to various applications.

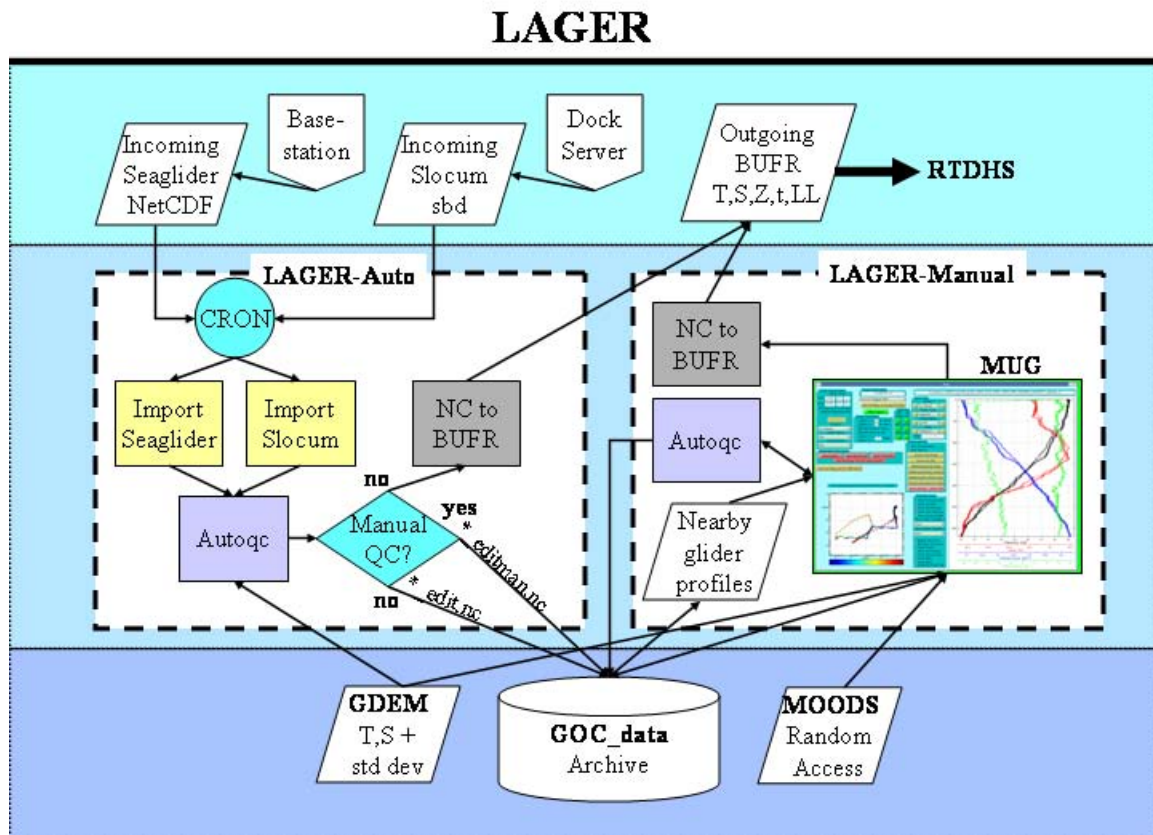
The bottom layer illustrates three databases used by LAGER. GDEM (Generalized Digital Environmental Model) is a three-dimensional monthly gridded global temperature and salinity mean and standard deviation climatology developed by the NOO. MOODS (Master Oceanographic Observation Data Set) is the NOO global archive of in-situ temperature and salinity observations. In the form used here, each temperature and salinity profile has been interpolated to the GDEM standard depths and stored in a random access binary file for rapid access.

The center layer illustrates the LAGER automated quality control analysis on the left and the manual quality control and editing on the right.

The automated QC is restarted at predefined time intervals (e.g., every 15 minutes) according to the cron scripts. The cron is a Unix/Linux system daemon that executes scheduled commands. The cron starts the master automated QC script (`$glider_qc_home/SCRIPTS/glider_import_qc.m` which performs the operations illustrated in the box labeled "LAGER-Auto" in Figure 1. This script first imports both the raw Seaglider NetCDF files and the Slocum sbd files. To import Seaglider files, the file name is changed to match the Lager file naming convention (Section 3.3 below) and

the original and renamed files are put into the data archive (labeled GOC\_data Archive in Figure 1). To import a Slocum file requires several steps discussed in Section 4.2 below. Primarily, this entails converting a Slocum sbd file into a file similar to a Seaglider NetCDF file, including changes to variable names and units. The multi-yoyo Slocum profiles are split into a sequence of descent-ascent profile pairs and each pair is written to a separate Seaglider-like NetCDF file. Both the original and the reformatted files are written to the glider data archive.

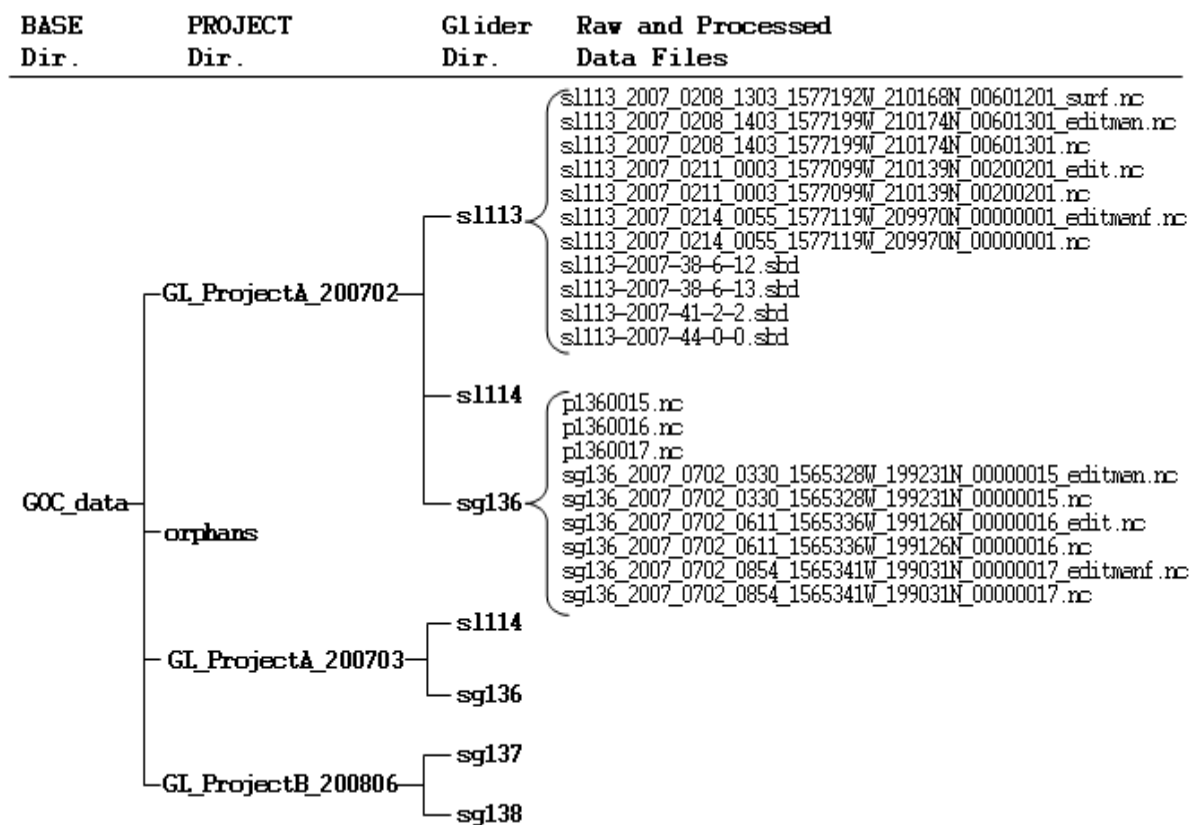
The reformatted files are read by the Autoqc program where the conductivity time series is corrected and each profile pair passes through several data quality tests. The results of the tests are stored as integer flags for each temperature and salinity observation. The overall quality of each profile is also assessed, resulting in a quality flag value for each whole temperature and salinity profile and a manual QC flag that specifies whether this entire data file should be sent to the manual editor for further processing. The corrected results and the flags are stored in a new file and saved in the glider data archive. Either the string "\_editman.nc" or "\_edit.nc" is appended to the edited output file depending on whether the file does or does not require further manual QC. If further manual QC is not required (depending on the manual QC flag), then it is also reformatted into BUFR format (box labeled "NC to BUFR" in Figure 1) and moved to the outgoing BUFR directory (top layer of Figure 1).



**Figure 12** Flow diagram of LAGER system.

## 3.2 Data base

Glider data files, both in their original un-edited format and in the reformatted standard format are stored in a three-level hierarchical flat file directory tree data base. A small example of directory and file structure of the database is shown in Figure 1. The path to the base directory is defined in the \$glider\_qc\_home/INFOFILES/glider\_qc\_info.m file as setup.base\_qc\_data\_directory. First-level subdirectories of the base directory are projects or area names defined in the \$glider\_qc\_home/INFOFILES/Areas file (see Section 2.9 above). The second level subdirectories are glider names which are also defined in the Areas file for each project. Within each glider subdirectory are both edited and unedited glider data files. The file names are discussed next Section 3.3.



**Figure 13** Example of the LAGER hierarchical glider data archive directory and data file structure.

## 3.3 File names

### 3.3.1 Seaglider initial

The initial Seaglider file names coming from the Basestation have the form:

pGGGDDDD.nc.

GGG is the three-digit seaglider platform number and DDDD is the dive number. The dive number can be reset by the operator at any time, but particularly at the start of a new project.

### 3.3.2 Slocum initial

The Slocum file names coming from the Dockserver have the form:

G-YYYY-JJJ-M-S.TTT.

GGGGG is the glider name (e.g., ru16 or sl113), YYYY is the four-digit year at the start of the mission, JJJ is the integer day of the year at the start of the mission, M is the integer mission number of the day (as many digits as required with no left-side zero fill), and S is the integer mission segment number (as many digits as required with no left-side zero fill). The file extension, indicated by TTT, is sbd for the short form, typically the type sent back over the Iridium satellite system, or is dbd for the long form, usually obtained only by direct connection to the glider after recovery. A file with mission segment 0 for a given mission number contains information about the variables contained in the file, and must be read by Webb software (dbd2asc, see Section 4.2 below) before any other segment of this mission.

### 3.3.3 LAGER NetCDF after import

After being imported to the LAGER system, data files have names of the form:

TTGGG\_YYYY\_mmdd\_hhmm\_oooooooB\_aaaaaaC\_dddddddd[E].nc

TT is the glider type (i.e., sg – Seaglider, sl – Slocum, sp – Spray). Glider names from the incoming file name that do not match the LAGER naming convention are converted to LAGER names on import (e.g., ru05 is converted to sl005) as discussed in Section 2.8 above. YYYY is the four-digit year, mm is the two-digit (zero-filled on left), dd is the two-digit day of the month (zero-filled on left), hh is the two-digit hour of the day (zero-filled on left), mm is the two-digit minute of the hour (zero-filled on left), oooooooo is the seven-digit longitude times 10000 (zero-filled on left), B is hemisphere (E or W), aaaaaa is the six-digit latitude times 10000 (zero-filled on left), and C is the hemisphere (N or S). The dddddddd field is an 8-digit (zero-filled on left) dive-related integer number. For Seaglider files, it has the form, 0000DDDD, where DDDD is the seaglider dive number described in Section 3.3.1 above. For Slocum files, it is constructed from MMMSSDD, where M is the three-digit (zero-filled on left) mission number from the original file name, SSS is the three-digit mission segment number (zero-filled on left) from the original file name, and DD is the two-digit yo pair number (zero-filled on left). Slocum files typically have several yo's (a dive descent and ascent pair). The original Slocum data files are split into separate yo pairs before being reformatted into LAGER NetCDF files. The [E] field is optional. The file names given to LAGER files immediately after being imported, but before being processed by the automated QC software do not use this field, except for one special case of Slocum files. If the initial Slocum file contains only surface observations obtained while the glider was transmitting data at the surface, then the [E] field is set to \_surf. Files of this type are never processed through the automated editor

software. Otherwise, for all types of gliders [E] is set to `_edit` after passing through the automated QC if the file does not require further manual QC, or [E] is set to `_editman` if further manual QC is required (as determined by the automated QC software). During manual QC using the mug gui-based manual editing software, the [E] field is set to `_editmanf` after being viewed, either with or without editing. After being accepted as complete by the mug editor operator (by sending to BUFR file creation), the `_editmanf` field is changed to `_edit`. At any given time there exists only one edited file type and it has the ending `_edit`, `_editman`, or `_editmanf`.

### 3.3.4 BUFR file

BUFR file names are nearly the same as those for the LAGER NetCDF data files. They have the form:

```
TTGGG_YYYY_mmdd_hhmm_oooooooB_aaaaaaC_ddddddddQ.bufr
```

All fields are the same as those discussed in Section 3.3.3, but with the edition of Q.bufr. The Q field is either u or d (no preceding underscore symbol) indicating an ascending profile or descending profile, respectively. The yo descent/ascent profile pair is split into separate profiles and each is written to a separate BUFR file. The file extension is .bufr.

## 4 Import Glider Observations

### 4.1 Seaglider

Seaglider NetCDF files with names like p1340098.nc (see Section 3.3.1 for file naming convention) are renamed (see the LAGER conventions for new names in Section 3.3.3) and modified by program `import_seaglider.m` to conform to the LAGER conventions. The Seaglider NetCDF data file format produced by the BaseStation system was upgraded near the beginning of the RIMPAC 2008 exercise. The upgrade included several new global attributes. The full set of global attributes with example field values is listed below with new attributes added by the upgrade preceded by an asterisk.

```
Conventions = "CF-1.0"
* instrumentid = "SG128"
  title = "SG128 NAVO"
  institution = "University Of Washington Applied Physics Lab"
  history = "Written Wed Jul 30 04:09:43 2008"
* dive_number = 239
* file_version = 2.f
* base_station_version = 2.3f
* file_data_type = "timeseries"
```

The upgrade also changed the names of many variables and value of the missing value attributes of variables. The listing in Appendix C shows the main variables (those accessed by LAGER) in the new format version together with attribute definitions. In Appendix D, the corresponding variables and attributes from the prior version format are

shown. The main differences between the two versions are that the missing value is "inf" in the old format and "nan" and variable names were changed as shown in the following listing:

<b>Old Format</b>	<b>New Format (Version 2.f)</b>
elaps_t_0000	eng_elaps_t_0000
elaps_t	eng_elaps_t
depth	eng_depth
head	eng_head
pitchAng	eng_pitchAng
rollAng	eng_rollAng
pitchCtl	eng_pitchCtl
rollCTL	eng_pitchCtl
rollCTL	eng_rollCTL
vdbCC	eng_vdbCC
redRef	wlbb2f_redRef
redCount	wlbb2f_redCount
blueRef	wlbb2f_blueRef
blueCount	wlbb2f_blueCount
fluorCount	wlbb2f_fluorCount
VFtemp	wlbb2f_VFtemp
depth_m	depth
lat	latitude
lon	longitude
Pressure	pressure
TempC_Cor	temp
cond_cor	conductivity
Salinity	salinity
SigmaT	sigma_t
TempC_Cor_pot	theta
Density	density
Density_pot	sigma_theta

In addition, two new variables, u\_da and v\_da, the east-west and north-south components of the vertically-averaged current, respectively, were added to the new format. The new variables names are used through the subsequent LAGER processing and file creation. When a file with the old format is read by program import\_seaglider.m, variable names, variable attributes, and Global attributes are converted to the new 2.f format.

The time variable used in the initial Seaglider files has units of integer seconds since January 1, 1970. The Slocum files use higher-precision decimal seconds since January 1, 1970. To allow a single variable in the LAGER file format that encompasses the time variable used by both gliders, a new variable, dtime, is added to the Seaglider NetCDF file by import\_seaglider.m. The dtime variable is a double precision (8 bytes) decimal number with units of decimal seconds since January 1, 1970. Similarly, a new double precision variable, dlog\_gps\_time, is also added that compliments the single-precision

integer time (seconds since January 1, 1970) variable that holds the time when GPS fixes were taken.

## 4.2 Slocum

The sbd (short binary data) files output by the Slocum Dock Server are imported to LAGER and reformatted into files similar to the imported Seaglider NetCDF files. The variables available in the sbd files vary greatly among gliders and glider missions. Variables sent back over the Iridium communications system are selected by the glider operators from a list of around 1200 possibilities. Even when a variable is sent back, its values may be all nans or all equal to a single constant value, depending upon various factors including hardware, software, or even human errors. This highly variable data set being returned from the Slocums poses a significant obstacle to the creation of Seaglider-like NetCDF files. Each Seaglider file contains two consecutive profiles from a single dive, a descending profile and a subsequent ascending profile. Data in a single sbd file from a single Slocum dive (period of time after the glider leaves the surface and when it re-surface to perform data transmission and GPS acquisition) may contain several yo-yos (descent-ascent pairs). In addition, profile observations might be made only on either the descent or ascent (usually not on the descent). To make a Seaglider-like file, the series of yo-yos must be split into separate descent-ascent pairs, with each pair written to a separate NetCDF file.

The LAGER program, `import_slocum.m`, performs the reformatting task. First, the initial Slocum sbd file is read by the compiled program from Webb Research Corporation, `dbd2asc`, which was installed in the `SLOCUM_BASE` subdirectory of `$glider_qc_home` (see Appendix A for directory listings). `dbd2asc` dumps the sbd file into an ascii file named `temporary_slocum_ascii_dump`. This file is read in by `import_slocum.m` as an multi-column `mxn` time series array, data, where `m` is the length of the time series and `n` is the number of columns. Each column holds the time series of one variable. The variable names, variable units, and variable size (number of bytes) are stored in arrays `variables`, `units`, and `var_size`, respectively.

The listing below shows the mapping of Slocum variable names into output Seaglider-like NetCDF file variable names. The column on the left lists the Seaglider names and the second column lists the original Slocum variable names. If there is more than one Slocum variable on the right, then the first one that is available, in the order shown, from the sbd file is used. All variables in the first set (down to `horz_speed_pitch_buoy_model`) are time series (functions of the output variable, time). The Slocum time has units of decimal seconds since January 1, 1970. To be compatible with the Seaglider NetCDF files, the time is stored as integer seconds in the output file in variable `time` and as a double precision read array in `dtime`. Presently, the depths stored in Slocum variable `m_depth` are not used because it often contains large gaps. Instead, depth is computed from pressure using the conversion algorithm, `sw_dpth`, in the Seawater Toolbox (see Section 2.4). Later versions of LAGER might use `m_depth`, depending on the results of further study.

The dead-reconned glider position is available in the Slocum variables `m_lon` and `m_lat` if they are sent back in the `sbd` file. If available these positions are stored in the output variables, `drlon` and `drlat`, which are not variables used in the NetCDF files made from Seaglider observations. If `m_lon` and `m_lat` are available, their positions are linearly stretched to match the GPS positions at the start and end of the dive, but only if the GPS positions `m_gps_lon` and `m_gps_lat` are available from the `sbd` file. These modified dead-reconned positions are stored in the output variables, `longitude` and `latitude`. If `m_lon` and `m_lat` are available but `m_gps_lon` and `m_gps_lat` are not available, then `m_lon` and `m_lat` are stored in `longitude` and `latitude`. If `m_gps_lon` and `m_gps_lat` are available, then the `latitude` and `longitude` output array values at each time are linearly interpolated from the GPS `longitude` and `latitude` at the beginning and ending of the dive. If the GPS positions do not span the time of the dive, then the `longitude` and `latitude` are set to a constant value if either the starting GPS position or the ending GPS position is available. If neither position is available, then the `latitude` and `longitude` arrays are set to all nans.

The vertical velocity of the glider is computed from the depth versus time and stored in `vert_speed_pitch_buoy_model`. The horizontal velocity is estimated from the `latitude` and `longitude` (computed as discussed above from dead-reconned and/or GPS positions) versus time.

Three sets of variables, GPS positions, altimeter depths, and vertically-averaged water velocity components, are stored in time series arrays and output in variable with names that are not used in the NetCDF files made from Seaglider observations. The length of the time series is, in general, different for each of these variables. For each, the time and position, and possibly the depth, are obtained at the time of each observation from the `time`, `longitude`, `latitude`, and `depth` output variables. In the next version of LAGER, these variables will be renamed and revised to match similar variables obtained from the Seaglider.

The remaining variables available from the Slocum `sbd` file, other than those already discussed or used in forming output variables already discussed, are output to the NetCDF file with their original `sbd` file variable names and units. This final set of variable is output as single precision real variables except for the time-related variables, `sci_ctd41cp_timestamp`, `sci_water_timestamp`, and `sci_m_present_time`, which are output as double precision variables.

The time series are divided into a sequence of consecutive descent/ascent profile pairs, and each pair is written to a separate output NetCDF file. The output file name for each file (see Section 3.3.3) contains the `latitude`, `longitude`, `date`, and `time` computed for each profile pair as the

<code>time(time)</code>	<code>m_present_time</code>
<code>dtime(dtime)</code>	<code>m_present_time</code>
<code>pressure(time)</code>	1. <code>sci_water_pressure</code>

	2. m_water_pressure
	3. m_pressure
	4. m_depth + latitude
temp(time)	1. sci_water_temp 2. m_water_temp
conductivity(time)	1. sci_water_cond 2. water_cond
depth(time)	1. m_depth (presently not using m_depth) 2. pressure + latitude (available/selected pressure above)
longitude(time)	m_lon and/or m_gps_lon
latitude(time)	m_lat and/or m_gps_lat
vert_speed_pitch_buoy_model(time)	Change in depth versus change in time.
horz_speed_pitch_buoy_model(time)	Change in position versus change in time.
if available, the following variables are output.	
drlon(time)	m_lon
drlat(time)	m_lat
log_gps_time(log_gps_time)	m_present_time
dlog_gps_time(dlog_gps_time)	m_present_time
gps_lon(log_gps_time)	m_gps_lon
gps_lat(log_gps_time)	m_gps_lat
altimeter_time(altimeter_time)	m_present_time
altimeter_depth(altimeter_time)	m_water_depth
altimeter_sensor_depth(altimeter_time)	depth
altimeter_lon(altimeter_time)	longitude
altimeter_lat(altimeter_time)	latitude
water_velocity_time	m_present_time
water_velocity_u	m_final_water_vx m_water_vx
water_velocity_v	m_final_water_vy m_water-vy
water_velocity_lon	longitude
water_velocity_lat	latitude

Other variables if available, are output keeping the original variable name and units.

Double precision real  
sci\_ctd41cp\_timestamp

sci\_water\_timestamp  
sci\_m\_present\_time  
Single precision real  
All other variables including control variable and optical parameters

## 5 Automated QC

### 5.1 *Quality Control tests and flag setting at each depth*

Many of the quality control tests implemented in LAGER were derived from tests presented in several publications, including UNESCO (1990), Boyer and Levitus (1994), Maudire (1994), Levitus (2005), Ingleby and Huddleston (2007), Schmid et al. (2007), Gronell and Wijffels (2008). To these tests, several glider-specific tests were added to detect and flag specific known types of bad behavior exhibited by either specific brands of gliders or by all types of gliders. In most cases, the glider-specific tests are functions of the vertical velocity of the glider which is employed as a substitute for the more-difficult-to-determine total speed of the glider through the water. All of the QC tests in LAGER are independent of geographic location and time of year except those that compare observations to the GDEM climatology, and the tests are almost independent of depth except in cases where different critical test values are used in two different depth ranges. The universal character of the tests weakens their capability to detect erroneous anomalies. In future versions of LAGER, we expect to use critical test values determined for some regions where large amounts of historical glider data are available.

Temperature flags ( $\text{temp\_flag}(i)$ ) and salinity flags ( $\text{salt\_flag}(i)$ ) at each depth ( $Z(i)$ ), where  $i$  is the depth index, are initially set to zero at each depth. Tests are performed on observed temperature values ( $T(i)$ ) prior to any tests performed on salinity values ( $S(i)$ ). In addition, one test (spike) is performed on conductivity ( $C(i)$ ). If a test is failed at a depth with index  $i$ , then the corresponding flag at the given index is set to the failure flag value for that test. The sequence of tests is described next.

#### 5.1.1 Depth check T and S

$\text{temp\_flag}(i) = 1$  and  $\text{salt\_flag}(i) = 1$  if  $Z(i) < 0$  m (surface).

#### 5.1.2 Global bounds check T

$\text{temp\_flag}(i) = 2$  if  $T(i) < -2.5^\circ \text{C}$  or  $T(i) > 43^\circ \text{C}$ .

For comparison, critical values are  $-2.5^\circ \text{C}$  and  $40^\circ \text{C}$  in Schmid et al. (2007).

#### 5.1.3 Comparison to GDEM T

GDEMv3.0 is the navy's standard monthly ocean temperature and salinity climatology (ref). It is global, monthly, has a 0.25 degree geographic latitude and longitude resolution, and is defined at 78 standard depths from the surface to 6600 m depth. For each observation profile, the GDEM temperature ( $T_g$ ) and temperature standard deviation ( $T_{gstd}$ ) profiles from the nearest location and month are extracted and interpolated to the depths of the observed profile.

temp\_flag(i) = 3 if  $|(T(i)-T_g(i))/T_{gstd}(i)| > 5$ .

#### 5.1.4 Spike test T

temp\_flag(i) = 4 if  $|T(i)-(T(i+1)+T(i-1))/2| - |T(i+1)-T(i-1)| > K$ , where  $K = 2^\circ \text{C}$  ( $Z(i) < 500 \text{ m}$ ) or  $K = 1^\circ \text{C}$  ( $Z(i) \geq 500 \text{ m}$ ).

For comparison, critical values are  $6^\circ \text{C}$  and  $2^\circ \text{C}$  in the same ranges in Schmid et al. (2007).

#### 5.1.5 Gradient test T

temp\_flag(i) = 5 and temp\_flag(i+1) = 5 if  $|(T(i+1)-T(i))/(Z(i+1)-Z(i))| > K$  where  $K = 4^\circ \text{C/m}$  ( $Z(i+1) < 500 \text{ m}$ ) or  $K = 1^\circ \text{C/m}$  ( $Z(i+1) > 500 \text{ m}$ ).

For comparison, in Schmid et al. (2007), the spike test critical value is  $|T(i)-(T(i+1)+T(i-1))/2| > K$ , where  $K = 9^\circ \text{C}$  ( $Z(i) < 500 \text{ m}$ ) or  $K = 6^\circ \text{C}$  ( $Z(i) \geq 500 \text{ m}$ ).

#### 5.1.6 Spike test C

salt\_flag(i) = 4 if  $|C(i)-(C(i+1)+C(i-1))/2| - |C(i+1)-C(i-1)| > K$ , where  $K = 0.02^\circ \text{S/m}$  ( $Z(i) < 500 \text{ m}$ ) or  $K = 0.01^\circ \text{S/m}$  ( $Z(i) \geq 500 \text{ m}$ ).

#### 5.1.7 Vertical velocity test

The depth,  $Z(i)$ , is first interpolated to a one-second time grid and smoothed with an 11-point running average. The vertical velocity at the center of each one-second time interval is computed as by centered differences of the smoothed depth divided by the time interval. The resulting vertical velocity is then interpolated back to the original time grid to form the  $W(i)$  series. The Seaglider times are reported as integers, resulting in a time uncertainty of 0.5 seconds. The smoothing is performed primarily to remove noise in the computed vertical velocity resulting from the time truncation. The Slocum glider times are reported as double precision floating point numbers. In addition, the Slocum may report both a glider system clock time (variable name is `m_present_time`) and a CTD clock time (variable names, in order of preference, are either `sci_ctd4lcp_timestamp`, `sci_water_timestamp`, or `sci_m_present_time`). If a CTD clock time is present, then it is used both in the calculation of the vertical velocity and in the correction of the conductivity time series (see Section 5.3 below). The seaglider also reports back a vertical velocity having the variable name, `vert_speed_pitch_buoy_model`. This variable is not used in LAGER processing because its values sometimes suddenly become constant (mainly zero), particularly near the surface during the glider ascent. The smoothed vertical velocity computed by LAGER ( $W$ ) is written to the output NetCDF glider data file as variable name `vert_speed_depth_time`.

##### 5.1.7.1 Slocum

salt\_flag(i) = 10 if  $|W(i)| < 2 \text{ cm/s}$ .

##### 5.1.7.2 Seaglider

salt\_flag(i) = 10 if  $|W(i)| < 5 \text{ cm/s}$ .

If  $Z(i) > 10$  m and  $Z_{\max} - Z(i) > 10$ , where  $Z_{\max}$  is the maximum depth of the present dive, then  $Tflag = 10$  and  $Sflag = 10$  for depths from  $Z(i) - 2$  m to  $Z(i) + 2$  m, if  $|W(i)| < 3$  cm/s.

$temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$  if  $|W(i)| < 6$  cm/s and  $Z_{\max} - Z(i) < 10$  m, where  $Z_{\max}$  is the maximum depth of the present dive.

During the descent,  $temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$  at all depths  $Z_{crit} \leq Z(i) \leq Z_{\max}$ , where  $Z_{crit}$  is depth where  $|W| < 6$  cm/s in the depth range,  $Z_{\max} - Z(i) < 15$  m.

During the descent, for all  $Z(i)$  in the upper 3 m before  $W$  first exceeds 5 cm/s,  $temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$ .

During ascent, for all  $Z(i)$  in the upper 5 m after  $W$  falls below 5 cm/s,  $temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$ .

### 5.1.8 Surface Chop

#### 5.1.8.1 Slocum

$temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$  if  $Z(i) < 0.2$  m.

#### 5.1.8.2 Seaglider

$temp\_flag(i) = 20$  and  $salt\_flag(i) = 20$  if  $Z(i) < 2$  m.

### 5.1.9 Global bounds check S

Salinity is computed from the corrected conductivity (see thermal lag correction below) and then salinity is modified to produce a statically stable profile to produce the final salinity ( $S(i)$ ).

$salt\_flag(i) = 2$  if  $S(i) < 0$  psu or  $S(i) > 45$  psu.

For comparison, critical values are 0 psu and 41 psu in Schmid et al. (2007).

### 5.1.10 Comparison to GDEM, S

$salt\_flag(i) = 3$  if  $|(S(i) - S_g(i)) / S_{gstd}(i)| > 5$ .

### 5.1.11 Spike test S

$salt\_flag(i) = 4$  if  $|S(i) - (S(i+1) + S(i-1))/2| - |S(i+1) - S(i-1)| > K$ , where  $K = 0.25$  psu ( $Z(i) < 500$  m) or  $K = 0.125$  psu ( $Z(i) \geq 500$  m).

For comparison, critical values are 0.9 psu and 0.3 psu in the same ranges in Schmid et al. (2007).

### 5.1.12 Gradient test S

$salt\_flag(i) = 5$  and  $salt\_flag(i+1) = 5$  if  $|(S(i+1) - S(i)) / (S(i+1) - S(i))| > K$  where  $K = 0.2$  psu/m ( $S(i+1) < 500$  m) or  $K = 0.15$  psu/m ( $Z(i+1) > 500$  m).

For comparison, in Schmid et al. (2007), the spike test critical value is  $|S(i) - (S(i+1) + S(i-1))/2| > K$ , where  $K = 1.5$  psu ( $Z(i) < 500$  m) or  $K = 0.5$  psu ( $Z(i) \geq 500$  m).

## 5.2 Manual Flags

Manual flags are set at each depth in the MUG GUI manual profile editor. If a file does not pass through the MUG editor, then all manual editor flags are set to zero.

### 5.2.1 temperature

<code>manual_temp_flag(i) = 0</code>	no change of automated flag (use <code>temp_flag</code> )
<code>manual_temp_flag(i) = 1</code>	manually changed value (now good, ignore <code>temp_flag(i)</code> )
<code>manual_temp_flag(i) = 2</code>	interpolated value (now good, ignore <code>temp_flag(i)</code> )
<code>manual_temp_flag(i) = 3</code>	manually set value to bad (bad now, ignore <code>temp_flag(i)</code> )

### 5.2.2 salinity

<code>manual_salt_flag(i) = 0</code>	no change of automated flag (use <code>salt_flag</code> )
<code>manual_salt_flag(i) = 1</code>	manually changed value (now good, ignore <code>salt_flag(i)</code> )
<code>manual_salt_flag(i) = 2</code>	interpolated value (now good, ignore <code>salt_flag(i)</code> )
<code>manual_salt_flag(i) = 3</code>	manually set value to bad (bad now, ignore <code>salt_flag(i)</code> )

## 5.3 Overall quality flags

### 5.3.1 keeptemp\_flag

The flag for the quality of the entire temperature profile is `keeptemp_flag`. Typically, there is one ascending profile and one descending profile in each file and the `keeptemp_flag` variable is a two-value integer array with one value for each profile. `keeptemp_flag = 1` if the temperature profile is to be kept. `keeptemp_flag = 0` if at least 30 profile depth or 30% of depths have a `temp_flag > 0` but not counting `temp_flag` values = 20.

### 5.3.2 keepsalt\_flag

The flag for the quality of the entire salinity profile is `keepsalt_flag`. Typically, there is one ascending profile and one descending profile in each file and the `keepsalt_flag` variable is a two-value integer array with one value for each profile. `keepsalt_flag = 1` if the salinity profile is to be kept. `keepsalt_flag = 0` if at least 30 profile depth or 30% of depths have a `salt_flag > 0` but not counting `salt_flag` values = 20.

### 5.3.3 needs\_manual\_editing\_flag

A `needs_manual_editing_flag` is assigned in each file to each ascending or descending profile (each value summarizes quality of the combined temperature and salinity profile). `needs_manual_editing_flag = 0` indicates that manual editing is not required. `needs_manual_editing_flag = 1` indicates that the file containing this profile should be sent to the manual editor for further examination even if the `needs_manual_editing_flag` for the other profile (if it exists) in the file is set to 0 (doesn't need manual editing). `needs_manual_editing_flag = 1` if:

1. at least one temp\_flag or salt\_flag = 3 (failed GDEM comparison test, 5.1.2 and 5.1.10 above).
2. at least 15 or 15% of the depths have temp\_flag > 0, but not including temp\_flag = 20.
3. at least 15 or 15% of the depths have salt\_flag > 0, but not including salt\_flag = 20.
4. at least 15 or 15% of the depths of the original, uncorrected salinity were flagged as unstable.
5. for Slocum profiles only, at least one depth gap between consecutive observations was >= 10 m or two gaps >= 6 m.

## 5.4 Conductivity correction

Salinity is computed from the conductivity, temperature, and pressure measured on the glider by a non-pumped CTD (SBE 41) manufactured by Seabird Electronics, Inc. Calculation of accurate salinity requires corrections for spatial measurement offsets of the sensors, for differences in the sensor response times, and for the thermal inertia of the conductivity cell. With the non-pumped CTD, the speed of flow through the conductivity cell depends upon the speed of the glider, making the thermal inertial correction speed-dependent. The LAGER Version 1 software performs only a speed-independent correction which is adequate for most cases, but it appears to over-correct when the glider vertical velocity is greater than 20 cm/s. The coefficients used in the correction algorithm were obtained for the Slocum from research performed by Kerfoot et al. (2006). Coefficients were not available for the Seaglider, so they were computed as part of the LAGER development from several hundred Seaglider profiles already measured by NOO.

A discrete time-domain recursive filter was developed by Lueck and Piclo (1990) for the conductivity correction in terms of temperatures given by

$$C_T(n) = -bC_T(n-1) + \gamma\sigma[T(n) - T(n-1)] \quad (1)$$

where

$$a = 4f_n\sigma\varphi^{-1}(1 + 4f_n\varphi^{-1})^{-1} \quad (2)$$

and

$$b = 1 - 2a\sigma^{-1}. \quad (3)$$

In these equations,  $n$  is the observation index,  $T$  is the measured temperature,  $f_n$  is the sample Nyquist frequency, and  $\gamma$  is the conductivity change due to temperature while holding salinity and pressure constant, i.e.,  $\partial C / \partial t|_{s,p}$ . The response of the measured conductivity has magnitude  $\sigma$  and e-folding time scale  $\varphi^{-1}$  of the temperature error.

Several approaches have been used to determine the coefficients  $\sigma$  and  $\varphi$  for various CTD models. Morison et al. (1994) determined the coefficients for the Sea-Bird SBE-9 CTD that sampled at 24 Hz and where the flow through the sensor was pumped at a constant rate of about  $1.75 \text{ m s}^{-1}$ . Their approach to determining the coefficients compared up cast and down cast profiles after correcting the profiles using Equations (1-3). The best set of coefficients was chosen as those that produced the minimum difference in T/S diagrams between the up and down casts. The approach is based on the assumption that the change

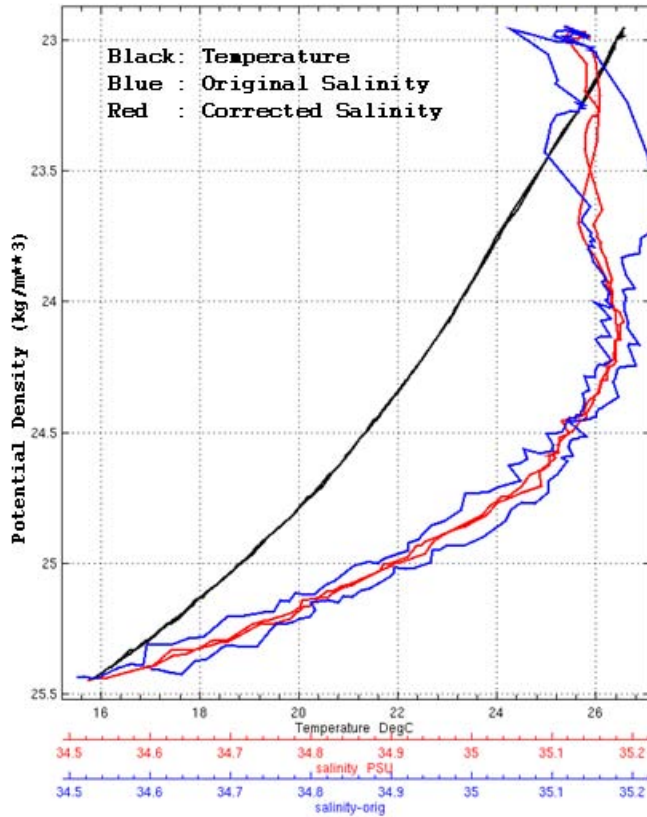
in the T/S relationship is small between the up and down casts. They also combined their results with several other studies that used both pumped and un-pumped Sea-Bird conductivity cells to determine  $\sigma$  and  $\phi$  as functions of flow rate through the cell. Their equations for the Sea-Bird cell are

$$\sigma = 0.0264V^{-1} + 0.0135 \quad (4)$$

and

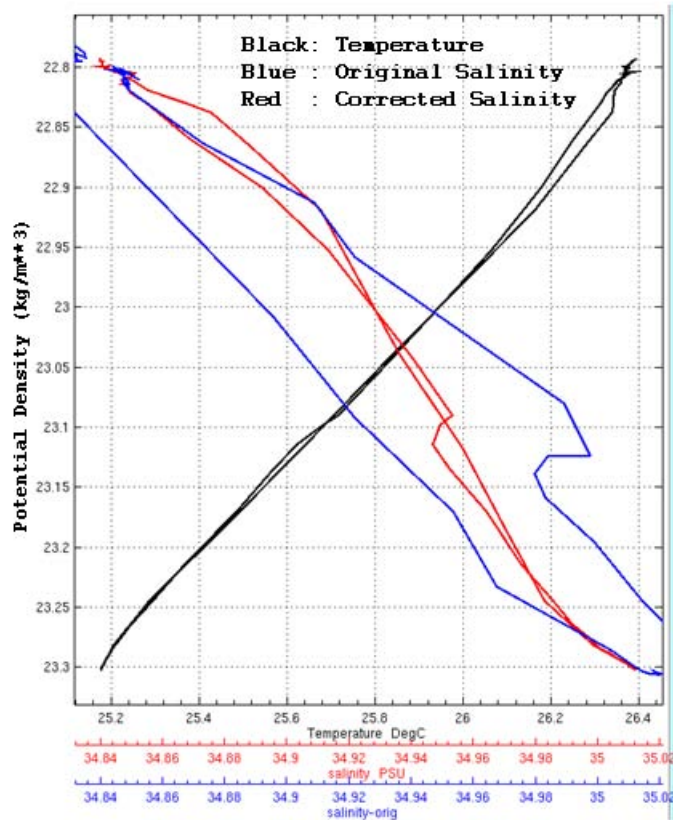
$$\phi^{-1} = 2.7858 V^{-1/2} + 7.1499, \quad (5)$$

where  $V$  is the velocity through the conductivity cell in units of  $\text{m s}^{-1}$ . Kerfoot et al. (2006) determined these coefficients for the Slocum Sea-Bird 41cp CTD using essentially the same technique used by Morison et al. (1994). They obtained  $\sigma = 0.13$  and  $\phi^{-1} = 25.5$ . The flow velocities computed by substituting these values into Equations (4) and (5) are  $22.7 \text{ cm s}^{-1}$  and  $2.3 \text{ cm s}^{-1}$ , respectively. These two inconsistent velocities indicate that the Slocum results do not fit within those presented by Morison et al. (1994). Glider velocities are highly variable; the ascent and descent velocities are often a factor of two different. In addition, the speed of flow through the conductivity cells is much lower than the speed of flow around the cell. For example, Morison et al. (1994) estimated that the flow speed through the Sea-Bird microconductivity cell was  $10 \text{ cm s}^{-1}$  when the CTD was being lowered at a rate of  $36 \text{ cm s}^{-1}$ . One of the goals of LAGER project is to eventually determine coefficients to provide velocity-dependent conductivity corrections for the Slocum, Spray, and Seaglider CTDs. However, the most viable approach compares consecutive descent and ascent profiles, whereas the Spray glider only makes observations on the ascent and the Slocum typically (under NOO operational conditions) makes observations only on the descent. Only the Seaglider samples on both the ascent and descent.



**Figure 14** Seaglider descending and ascending profiles of salinity and temperature versus potential density. The black lines show the temperature profiles and the blue lines show the salinity prior to correction of conductivity for thermal-lag effects. The red lines are the descending and ascending salinity after correction of the conductivity.

The SLOCUM conductivity measurements are presently corrected in LAGER using Equations (1-3) and the (average velocity-independent) coefficients determined by Kerfoot et al. (2006). The Seaglider coefficients were determined as part of the LAGER project from several hundred profiles from four different gliders (sg128, sg136, sg137, and sg138) using a technique similar to that used by Morison et al. (1994) and Kerfoot et al. (2006). The average velocity-independent Seaglider coefficients are  $\sigma = 0.1$  and  $\sigma = 31$ . An example of the salinity profiles before and after correction of the conductivity for the Seaglider is shown in Figure 3 and for the Slocum is shown in Figure 4.



**Figure 15** Same as Figure 3 except for Slocum.

## 5.5 Static stability

After correction of the thermal inertial of the conductivity cells, the resulting computed salinity is modified to remove density inversions. Density inversions can be caused by temperature inversions, where temperature increases with increasing depth, by salinity decreasing with increasing depth, or both. The correction algorithm used in LAGER assumes that the instability is due only to salinity. This assumption is justified due to the difficulty in correcting the conductivity of a non-pumped CTD, particularly when the sampling rate is often only 1 sample each 2 to 30 seconds. In addition, after examination of many glider profiles, no systematic tendency for the temperature profile to exhibit erroneous thermal inversion was revealed. However, in a recent exercise in shallow water near the Hawaiian Islands, many small thermal inversions were observed that were apparently the result of breaking internal waves induced by strong vertical shears from large-amplitude tides. Temporary instabilities such as these are probably real and should not be removed by modifying the salinity. However, the LAGER system presently does not have an algorithm to distinguish between true instabilities and those due to inaccurate observations.

One of the goals of profile stabilization is to localize the modification to the segment of the profile exhibiting the instability. Jackett and McDougall (1995) developed a method to correct instabilities with minimal adjustment of both the temperature and salinity

profiles as a constrained weighted least-squares analysis. The approach used in LAGER is a simpler iterative algorithm that produces localized adjustments to the salinity.

The LAGER algorithm adjusts the salinity to obtain a squared Brunt-Väisälä (BV) frequency,  $N^2$ , greater than a specified lower bound,  $N_{\min}^2$ , where

$$N^2 = \frac{g}{\rho_\theta} \frac{d\rho_\theta}{dz}, \quad (1)$$

$g$  is the gravitational acceleration,  $\rho_\theta$  is the potential density referred to the insitu pressure,  $p$ , and  $z$  is the depth (increasing downward here). Equation (1) can be rewritten as

$$N^2 = g \left[ \beta(p) \frac{dS}{dz} - \alpha(p) \frac{d\theta}{dz} \right] \quad (2)$$

where  $\theta$  is the potential density (referred to insitu pressure,  $p$ ),  $S$  is the salinity, and  $\alpha$  and  $\beta$  are the thermal expansion and saline contraction coefficients

$$\alpha = -\frac{1}{\rho} \frac{\partial \rho}{\partial \theta}, \quad \beta = \frac{1}{\rho} \frac{\partial \rho}{\partial S}. \quad (3)$$

Since only the salinity is to be adjusted, then the salinity vertical gradient,  $Q$ , that produces minimal BV frequency,  $N_{\min}^2$ , is

$$Q = \left. \frac{dS}{dz} \right|_{\min} = \left[ \frac{N_{\min}^2}{g} - A \right] / \beta \quad (4)$$

and

$$A = \alpha \frac{d\theta}{dz} \quad (5)$$

is a constant at each profile depth.

The salinity adjustment algorithm uses the discrete values,  $S_i$ ,  $\theta_i$ ,  $p_i$ ,  $\alpha_i$ ,  $\beta_i$ , at depths  $z_i$ ,  $i=1, n$ , where  $n$  is the number observation depths in the profile. First,  $A_j$ ,  $Q_j$  are computed at the mid-depth,  $z_j = (z_i + z_{i+1})/2$ , ( $j$  is the index for quantities between indexes  $i$  and  $i+1$ ) between each pair of consecutive observations. The thermal expansion and saline contraction coefficients are computed using the MATLAB functions, `sw_alpha` and `sw_beta`, respectively, contained in the SEAWATER Toolkit developed by Phil Morgan of the CSIRO Marine Research. The algorithms for each routine are based on McDougall (1987).

A salinity correction anomaly,  $s_i^k$  and  $s_{i+1}^k$ , for each depth index,  $i$ , is updated at each iteration,  $k$ , only if the minimum salinity gradient required for stability,  $Q_j$ , is greater than the salinity gradient of the  $k-1$  iteration, i.e.,

$$R_j^{k-1} = (S_{i+1}^{k-1} - S_i^{k-1}) / (p_{i+1} - p_i) < Q_j. \quad (6)$$

The salinity correction anomaly is initialized as all zeros before the first iteration and then is updated at all depths,  $i$ , where (6) is true by first applying

$$s_i^k = s_i^{k-1} - f(Q_j - R_j^{k-1})(p_{i+1} - p_i) / 2 \quad (7)$$

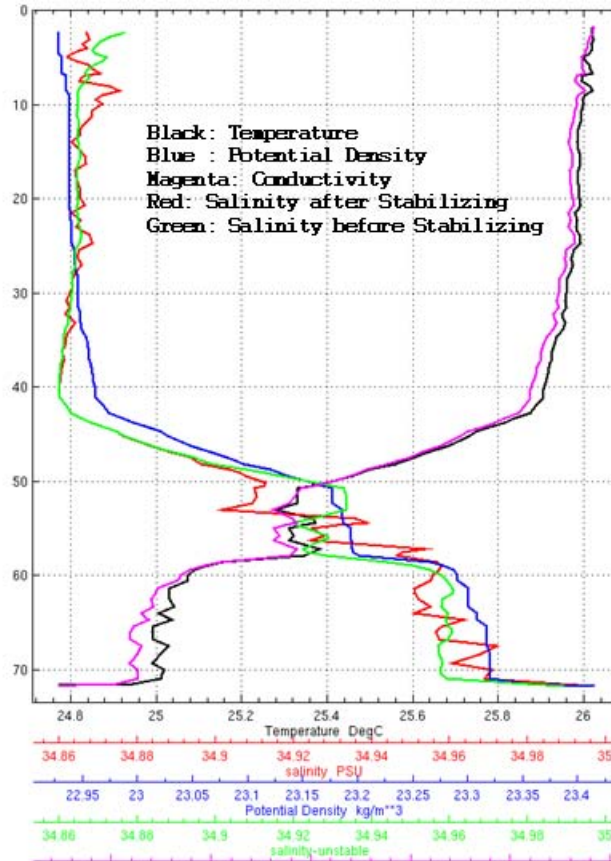
and then by applying

$$s_{i+1}^k = s_{i+1}^{k-1} + f(Q_j - R_j^{k-1})(p_{i+1} - p_i)/2, \quad (8)$$

where  $f = 0.75$  is a constant that causes only a fraction of the correction to be applied at each iteration. After each iteration, the anomaly is slightly smoothed using a running three-point filter with weights (0.01, 0.98, 0.01) and then is added to the initial salinity,

$$S_i^k = S_i^0 + s_i^k \quad (9)$$

at all depths. The iteration is continued until all instabilities are removed. This procedure tends to contain the correction within a small region around the instability as demonstrated in Figure 3, obtained from the MUG (the LAGER manual editor). Figure 3 shows a profile measured by a glider over the Penguin Banks southeast of Oahu in a regime of strong tides. The temperature inversions that occur over much of the depth range are corroborated by similar inversion of the independently-measured conductivity, and may be due to breaking internal waves resulting from strong tidally-induced vertical shears. The stable profile of potential density was obtained by applying the stabilizing algorithm discussed above to the salinity profile. In this special case, the true ocean density might be temporarily unstable and stabilization by LAGER might not be required.



**Figure 16** Profile showing changes in salinity profile performed to stability density.

## 6 BUFR File Generation

Version 000351 of the ECMWF BUFRDC software for encoding and decoding WMO FM-94 BUFR (Binary Universal Form for Representation of Meteorological data) messages was obtained from <http://www.ecmwf.int/products/data/software/bufr.html>.

The BUFR User's Guide and BUFR Reference Manual as well as several other documents including recent additions or changes awaiting validation are also available at this site. LAGER uses BUFRDC compiled for a Linux computer and Gnu project g77 and gcc compilers.

The LAGER software for converting edited glider data NetCDF files to BUFR files is called `glider_nc_to_bufr.f`. It is written in Fortran 77, compatible with the g77 compiler. It is designed to produce a BUFR message similar to that produced by a Perl script (`ArgoNetCDFp2BUFR_v2.pl`) supplied to this project by NOO which generates an Argo BUFR message from an Argo NetCDF mono-profile file.

A BUFR message is generated by making the following call to a MATLAB function:

```
make_bufr_file(nc_file_name,bufr_tables_directory, ...  
              bufr_executables_directory, going_to_rtdhs_dir)  
where,  
    nc_file_name =          full path and file name of the NetCDF file to read  
    bufr_tables_directory = directory containing BUFR tables  
    bufr_executables_directory = directory containing the BUFR conversion software  
    going_to_rtdhs_dir =    output directory for BUFR message
```

The directories are defined in `$glider_qc_home/INFOFILES/glider_qc_info.m`. The `make_bufr_file` MATLAB function calls a C-Shell script, `glider_nc_to_bufr.com`, that dumps the input glider data NetCDF file using `ncdump` and pipes the ascii output directly to the Fortran executable compiled from `glider_nc_to_bufr.f` which reads the ascii dump from unit 5 (standard input).

The Fortran program `glider_nc_to_bufr` reads the following variables from the ascii dump of the input glider data Netcdf file: salinity, longitude, latitude, depth, temp, time, temp\_flag\_index, salt\_flag\_index, prof\_start\_index, prof\_end\_index, temp\_flag, salt\_flag, keeptemp\_flag, keepsalt\_flag, manual\_temp\_flag, manual\_salt\_flag. The glider name and the type of glider (Seaglider, Slocum, or Spray) are determined from the first five characters of the input NetCDF file name. The input data arrays are split into separate ascending and descending profiles using the indexing information in arrays `prof_start_index` and `prof_end_index`.

The output BUFR message contains two sets of geographic position and time information. The first set consists of single values of latitude, longitude, and date/time at the profile center depth, representing the location and time of the entire profile. To compute these values, the center depth of the profile is first determined as one half the sum of the top depth plus the bottom depth. The nearest profile depth to this center depth is determined and the full-profile latitude, longitude, and date/time are set to the value of these variables at this same depth. The second set of positions and date/time are output in the BUFR message together with each observation depth, temperature, salinity, and data quality flag.

The LAGER data quality flags are converted to a single temperature GTSPQ QC code and a single salinity qc code for each observation. The global GTSPQ quality flags are listed in BUFR Code Table 033050:

- 0 Unqualified
- 1 Correct value (all checks passed)
- 2 Probably good but value inconsistent with statistics (differs from Climatology)
- 3 Probably bad (spike, gradient,... if other tests passed).
- 4 Bad value, impossible value (out of scale, vertical instability, constant profile).
- 5 Value modified during quality control.
- 6-7 Reserved.
- 8 Interpolated value.
- 9-14 Reserved.
- 15 Missing value.

The LAGER data quality flags are converted to GTSPQ QC codes in the following manner.

For each observation index, i, the following table lists the LAGER flags versus the GTSPQ QC codes (gtspt – temperature, gtsps – salinity).

LAGER QC flags	GTSPQ QC codes
Temperature:	
gtspt(i) = 1 if manual_temp_flag(i) = 0 and temp_flag(i) = 0 and keeptemp_flag = 1	
1. QC temperature flag changed during manual editing (manual_temp_flag(i) > 0)	
a. manual_temp_flag(i) = 1	gtspt(i) = 5 manually changed value
b. manual_temp_flag(i) = 2	gtspt(i) = 8 interpolated value
c. manual_temp_flag(i) = 3	gtspt(i) = 4 value set to bad
2. QC temperature flag not changed during manual editing (manual_temp_flag(i) = 0)	
a. temp_flag(i) = 1,2,10,or 20	gtspt(i) = 4
b. temp_flag(i) = 3	gtspt(i) = 2
c. temp_flag(i) = 4 or 5	gtspt(i) = 3
d. keeptemp_flag = 0 (whole prof)	gtspt(i) = 4 (overriding a,b, and c).

Salinity:

gtsps(i) = 1 if manual_salt_flag(i) = 0 and salt_flag(i) = 0 and keepsalt_flag = 1	
1. QC salinity flag changed during manual editing (manual_salt_flag(i) > 0)	
a. manual_salt_flag(i) = 1	gtsps(i) = 5
b. manual_salt_flag(i) = 2	gtsps(i) = 8
c. manual_salt_flag(i) = 3	gtsps(i) = 4
2. QC salinity flag not changed during manual editing (manual_salt_flag(i) = 0)	
a. salt_flag(i) = 1,2,6,7,10, or 20	gtsps(i) = 4
b. salt_flag(i) = 3	gtsps(i) = 2
c. salt_flag(i) = 4 or 5	gtsps(i) = 3
d. keepsalt_flag = 0 (whole prof)	gtsps(i) = 4 (overriding a,b, and c).

The BUFR descriptor reference identifies a BUFR Table reference. The latest BUFR tables are described in the BUFR Reference Manual (an unpublished Internal Report from ECMWF). A descriptor reference is associated with each value output in the BUFR message. The form of the reference is a 6 digit number of the form, FXY, where F is 1 digit, X is two digits and Y is three digits. The F value is 0 if this is an element descriptor and the X and Y values are entries in the BUFR Table B. F is 1 for a replication descriptor. F is 2 if the descriptor is one of the operators in Table C. F is 3 if the descriptor represents a sequence descriptor from Table D. To keep the BUFR message simple, all descriptors in the LAGER NetCDF to BUFR message conversion routine use Table B descriptors (except for one replication descriptor). Each value of X refers to a class of elements such as date, time, or position. Each value of Y is a descriptor of class. Although BUFR configuration allows the user to define local values for descriptors, only standard descriptors appearing in official WMO BUFR tables have been used in LAGER software.

The following section lists the descriptor references and values used in constructing a BUFR message by the LAGER routine `glider_nc_to_bufr.f`, in the order that they appear in the output message.

Description Reference	Value
1. Agency in charge of operating the observing platform, Code Table 001036	
001036	840006 (Naval Oceanographic Office)
2. Ship or mobile land station identifies (up to 9 chars)	
001011	5-char glider name, e.g., sg128
3. Observing platform manufacturer's model (up to 20 characters)	
001085	glider type, e.g., seaglider (or slocum or spray)
4. Direction of profile, Code Table 022056 (0 up, 1 down).	
022056	0 or 1
5. Date/time at profile mid-depth	
004001	4-digit year mid-depth, e.g., 2008
004002	month mid-depth, e.g., 9
004003	day of month mid-depth, e.g. 24
004004	hour of day mid-depth, e.g., 16
004005	minute of hour mid-depth, e.g. 59
004006	second of minute (integer) mid-depth, e.g., 34
6. High-resolution decimal latitude at mid-depth (positive north, negative south)	
005001	latitude mid-depth, e.g., 25.6226
7. High-resolution decimal longitude at mid-depth (positive east, negative west)	
006001	longitude mid-depth, e.g., -154.445
8. Replication sequence of 15 values, a delayed number of times (15 = number of variables to output for each observation depth)	
115000	nz number of observation depths
9. Extended delayed descriptor replication factor (needed if more than 128 obs.)	
031002	not set
10. This section consisting of 15 values is repeated for each observation (index i)	
004001	year(i)

004002	month(i)
004003	day(i)
004004	hour(i)
004005	minute(i)
004006	second(i) (integer)
005001	latitude(i)
006001	longitude(i)
007062	depth(i) in meters
022045	temperature(i) in degrees <b>Kelvin</b>
008080	code qualifier for GTSP quality class
033050	gtsspt(i) GTSP QC code for temperature(i)
022064	salinity(i) in psu
008080	code qualifier for GTSP quality class
033050	gtssps(i) GTSP QC code for salinity(i)

## 7 MUG (manual Utility for gliders)

MUG is a MATLAB-based GUI (Graphical User Interface) used to display and edit glider temperature and salinity profiles. It operates on glider data NetCDF files contained within subdirectories of the base QC data directory defined in \$glider\_qc\_home/INFOFILES/glider\_qc\_info.m. Later versions of MUG will also allow editing of optical parameter profiles, but presently, optical parameters can only be displayed. MUG was written using the MATLAB guide utility. Once the LAGER system has been set up as described in Section 2, MUG is initiated by first starting MATLAB (by typing MATLAB at the terminal prompt) and then typing MUG at the prompt in the MATLAB command window.

### 7.1 General operation

The mug GUI is shown in Figure 6 together with information boxes pointing to the primary control groups. All operations and plots remain contained within this one GUI. At GUI initiation, the map plot and the profile plot are empty, and all other edit boxes are at their blank or default state. To start an editing session, the user first completes the required operations within the "initial setup" (see Section 7.2) group of uicontrols (user interface controls). These controls allow the user to select the set profiles to edit according to their project, time period, glider type, and glider name. Once completed, the first profile pair from the first glider selected is displayed. In addition, the set of all positions occupied by that glider over the time period selected are displayed in the map plot. The display and editing always operate on the contents of a glider data NetCDF file. There is at most one descending profile and one ascending profile contained within a single data file. MUG operates on one file at a time, not one profile at a time, although various GUI options allow the separate up or down profile to be displayed or edited separately.

By selecting various options in the "profile setup" group of uicontrols (see Section 7.5), the profile plot can be modified to display the desired set of parameters (temperature, salinity, optical parameters, various glider control parameters, etc.). The profile plot can be zoomed to focus on a particular depth range, and profile values can be flagged or

changed using the buttons in the "zooming and editing" group of uicontrols (see Section 7.7). Data flagged by the LAGER automated QC software or previously flagged during manual editing within MUG can be displayed using operations available in the "flags and flagged data" group of uicontrols (see Section 7.8). The profile plot can be enhanced by adding other profile plots, such as from climatology, archived profiles, or surrounding glider profiles as underlays using controls in the "profile underlays" group of uicontrols (see Section 7.6).

Once the currently displayed profile pair of a data file has been reviewed and possibly edited, the next data file is selected using the controls in the "cycle through profiles and gliders" group of uicontrols (see Section 7.4). As the previous file leaves the display, its file name (still within the same subdirectory of the hierarchical glider data base) ending is changed from `_editman.nc` to `_editmanf.nc` to indicate that it has been manually edited or reviewed and is ready to be reformatted into a BUFR file and sent to the RTDHS. This same file can be reselected again later in the same editing session and re-edited if required.

After all or any portion of the set of glider files selected in the "initial setup" have been edited or reviewed, the editing session can be stopped using one of the options available in the "quit, exit, BUFR" group of uicontrols (see Section 7.9).

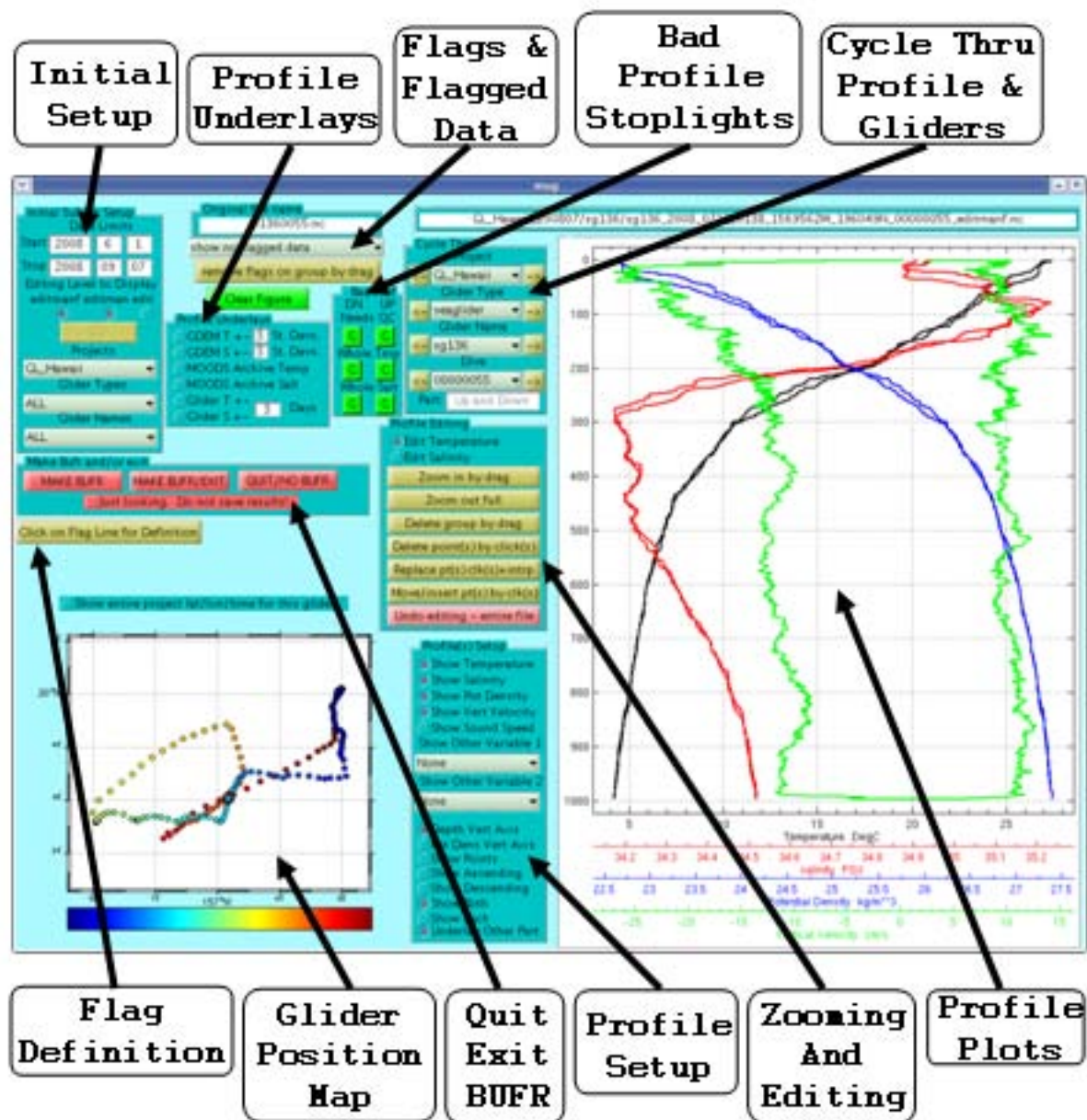


Figure 17 The MUG GUI.

## 7.2 Initial setup

After starting MUG, a sequence of operations must be completed in the "initial setup" group of uicontrols (Figure 7), in sequence from the top of the group to the bottom of the group, before profiles can be displayed and edited. These controls select the set of glider data files to be edited. At any time within an editing session, the controls within this group can be reset by the user to select a new set of glider files to edit.

At the top of the group are the year, month, and day of the **Start** and **Stop** time of the set of files to be edited. The default behavior of MUG is to set the **Start** value to today's

date minus three days and to set the **Stop** date to today's date plus one day. Each of these fields can be edited to change the dates as required. Inputting nonsensical values into these edit boxes results in display of ??, and requires correction before continuing.

The next set of uicontrols in this group consists of three radiobuttons that select the types of files to be edited based on the file name ending. All of any subset (but at least one) of files endings in **\_editmanf.nc**, **\_editman.nc**, or **\_edit.nc** can be selected. As discussed in Sections 3.1 and 3.3.3, files ending in **\_edit.nc** do not require further manual editing, files ending in **\_editman.nc** require further editing, and files ending in **\_editmanf.nc** have been edited or reviewed by MUG, but BUFR files have not yet been created from them. The default behavior of MUG is to select files ending in **editmanf.nc** and **\_editman.nc**.



**Figure 18** The "initial subset setup" group of uicontrols.

The next uicontrol of this group is the **GO** button which must be pushed to continue. If the date fields were modified and a return was not performed after the last change, then the **GO** button must be pushed twice to continue. Pushing the **GO** button causes MUG to look through the entire glider data directory tree for projects/areas, gliders, and data files within the selected date limits. This search operation requires several seconds, and employment of a **GO** button allows all changes to the date limits to be made before the search is started.

Next, the projects (or regions) are selected. Pushing the popup menu immediately below the **Projects** label displays a list of items which are project names containing gliders files within the selected date limits. These are the subdirectory names in the first level of the glider data base directory tree. At the top of the list is the extra item, **ALL**. Select either **ALL** to select all projects in the list, or select a single project.

Once a project is selected, the **Glider Types** popup menu is activated. Pushing the popup menu displays a list of all glider types (any subset of **Seaglider**, **Slocum**, or **Spray**) plus the item, **ALL**, at the top of the list. Press **ALL** to select all glider types or select one of the glider types.

Finally, the **Glider Names** popup menu at the bottom of this uicontrol group is activated. Pressing this popup menu displays a list of all gliders names in all of the projects and glider types already selected. In addition, the item, **ALL**, is added at the top of the list. Select **ALL** to include all glider names or select one glider name. Once selected, MUG starts reading in data to produce the map of the first selected glider positions and it read the data from the first selected glider file. This process can take several seconds. Once complete, the plots are drawn and the remaining uicontrol groups are activated and ready for use.

### 7.3 Cycling through profiles



**Figure 19** The "cycling through profiles and gliders" group of uicontrols.

The "cycling through profiles and gliders" group of uicontrols (Figure 8) enables the user to select a specific glider data file to be edited. There are four rows of values: **Projects**, **Glider Types**, **Glider Names**, and **Dives**. Pressing the popup menu for any of them displays the list of available values and allows any to be selected. The values available under each popup menu were determined from selections made during the initial setup (Section 7.2). For example, in the **Glider Type** popup menu there may be two values, **Seaglider** and **Slocum**. A specific value may be selected using the popup menu for any row or the previous or next value in the list can be selected by pressing the left or right arrow buttons (<- or ->) on the sides of the popup menus. The result from pressing the left arrow when at the beginning of a list or pressing the right arrow at the end of a list can still give somewhat unexpected results (to be fixed later :-). Generally, pressing the right arrow at the end of a list causes the list of values in the popup menu above it to be incremented and for the values in that row and all rows under it to be reset to the first value in the list.

At the bottom of the "cycle through" group of uicontrols is an edit box labeled **Part:**. Possible values in the edit box are **Up**, **Down**, or **Up and Down** depending upon the settings in the "profile setup" group of uicontrols (Section 7.4). If the **show both** radiobutton is on, then **Part:** will be **Down** if the file contains only a descending profile, or **Up** if the file contains only an ascending profile, or **Up and Down** if the file contains both a descending and ascending profile (in which case both profiles will be displayed at the same time). If the "profile setup" **show each** radiobutton is on, then only one profile from the file is displayed at a time, and the **Part:** value indicates whether it is the **Up** or **Down** profile. In addition, when the **show each** radiobutton is on, pressing the right arrow of the **Dive** row increments through each profile within a file before incrementing to the next file. When the **show both** radiobutton is on, pressing the right arrow on the **Dive** row increments the file and displays both profiles within the file (if the file contains both a descending and ascending profile).



**Figure 20** The "Bad Profile Stoplights" group of uicontrols.

As the profiles from each new glider data file are displayed, the set of colored buttons in the "Bad Profile Stoplights" group of uicontrols (Figure 9) indicates the present QC status of the ascending and descending profiles. For each button, a green color indicates that no overall quality flag has been set. A yellow color indicates that the profile is missing (i.e., the ascending profile is often not measured by the Slocum). A red color indicates that the overall bad quality flag has been set. This group contains six buttons in two columns and three rows. The first column indicates the overall quality of the descending profile and the second column for the ascending profile. The first row indicates the value of the `needs_manual_editing_flag` (see Section 5.3.3), with a red color meaning that the temperature or salinity profile requires manual editing and green meaning that manual editing is not required. The second row color is red if the `keeptemp_flag` (see Section 5.3.1) is set to 0 (entire temperature profile is bad) and green if it is set to 1 (good). The third row color is red if the `keepsalt_flag` (see Section 5.3.2) is set to 0 (entire salinity profile is bad) and green if it is set to 1 (good). The overall quality flags can be changed by pressing the buttons if the user does not agree with the settings or if the user changed the profile. Pressing a button changes it from green to red or from red to green. The `keeptemp_flag` and `keepsalt_flag` values are important because a profile marked red will be deleted from subsequently generated BUFR files.

## 7.4 Profile display options

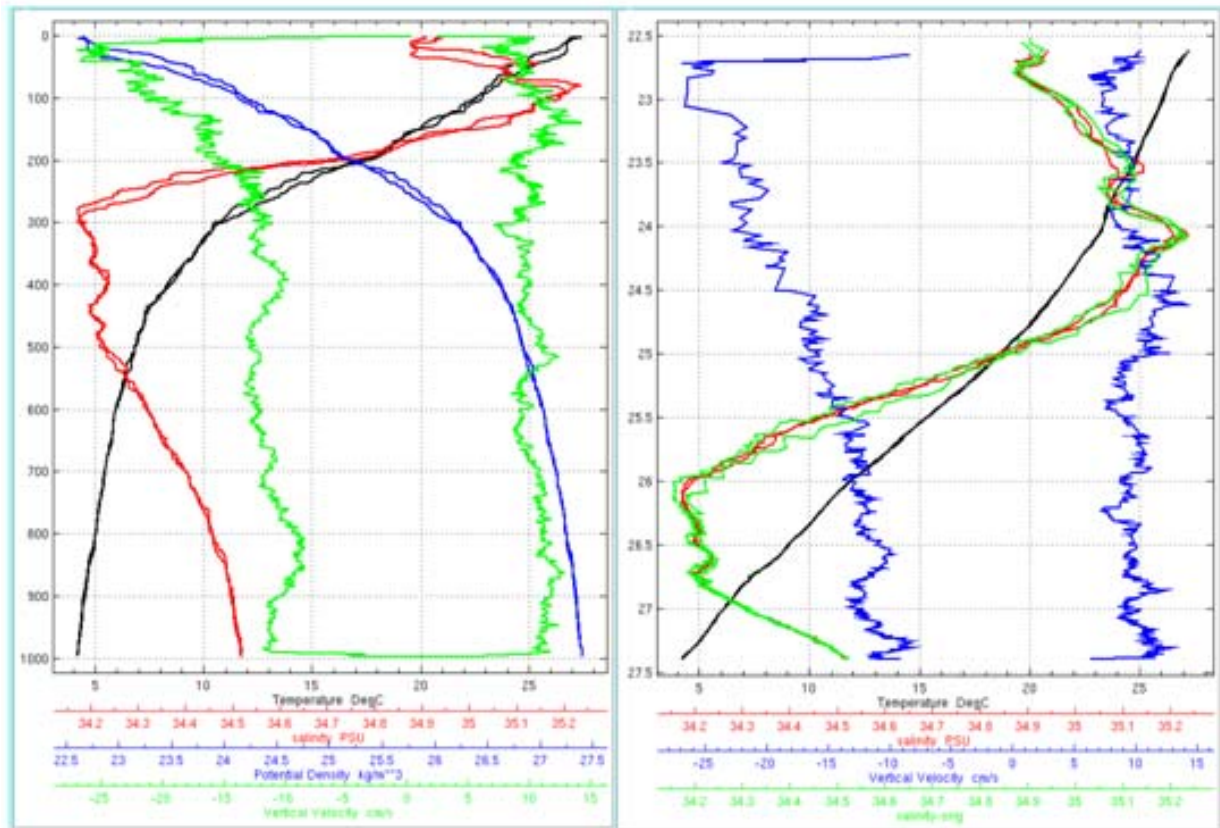
The "profile(s) setup" group of uicontrols (Figure 9) allows the user to control which variables are displayed as profiles and to also determine the style of the plot. Specific selection of temperature, salinity potential density, vertical velocity, and sound speed are allowed by turning on the **Show Temperature**, **Show Salinity**, **Show Pot Density**, **Show Vert Velocity**, and **Show Sound Speed** radio buttons, respectively. The default behavior is to show temperature, salinity, vertical velocity, a potential density and to use the reserved profile colors, black, red, green, and blue, respectively, for each. When one of these variables with reserved colors is not displayed, its color is used to display a non-reserved variable if selected. One or two other time series (or profile) variables available in the data file can also be selected for display using the **Show Other Variable 1** or **Show Other Variable 2** popup menus. These two popup menus list every time series (or profile) variable found in the data file (for example, salinity\_orig, cond\_orig, Scattering\_470, etc.). For each displayed profile, a horizontal axis labeled with the profile variable name and with in the same color as the profile is displayed at the bottom of the plot. A maximum of four horizontal axes can be displayed, although more than 4 profiles can be displayed.



Figure 21 "Profile display" options group of uicontrols.

Two types of vertical axes are available. The profile plot is drawn using a depth vertical axis if the **Depth Vert Axis** radiobutton is turned on and it is drawn with a potential density vertical axis if the **Pot Dens Vert Axis** radiobutton is on. Plotting with the potential density vertical axis is particularly useful when comparing the final salinity,

computed using the corrected conductivity (see Section 5.4) , to the original salinity (variable name salinity\_orig in the data file, computed using the uncorrected conductivity) as shown in Figure 11. In the short period between the descent and ascent, the watermass characteristics (potential temperature and salinity at constant potential density) should change little, so that the corrected salinity should be nearly identical versus potential density on the up and down casts. Activating the **Show Points** radiobutton causes the profiles to be drawn with a small circle symbol at each observation point.



**Figure 22** On the left, profiles of temperature, salinity, potential density, and vertical velocity using the default depth vertical axis setting (Depth Vert Axis radiobutton is turned on). On the right, profiles temperature, salinity, salinity\_orig, and vertical velocity are drawn versus potential density by turning on the Pot Dens Vert Axis radiobutton.

The bottom five radiobuttons control whether the ascending, descending, or both profiles are displayed. Activating the **Show Ascending** or **Show Descending** buttons causes only the ascending or descending profiles, respectively, to be drawn. Activating the **Show Each** button causes only one profile to be displayed. In this case, cycling through the profiles using the Dive right arrow button (in the "Cycle Thru" group of uicontrols) causes the descending profile to be drawn first, then, on the next increment (hitting the right arrow again), the ascending profile is drawn. Activating the **Show Both** button causes the ascending and descending profiles to be drawn simultaneously. If the **Underlay Other Part** button is activated, and either the **Show Ascending**, **Show Descending**, or

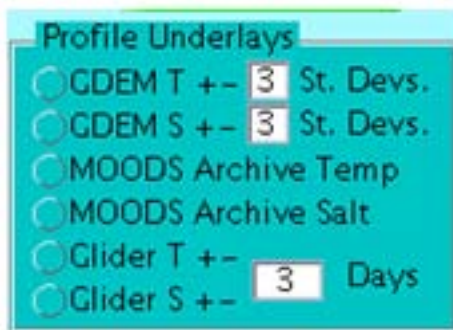
**Show Each** buttons is on, then the other (descending or ascending) profile in the file (if one exists) is also displayed, but as a cyan-colored line.

## 7.5 Underlay comparisons

Activating radiobuttons in the "profile underlays" group of uicontrols causes profiles from other sources to be drawn, for comparison, on the profile plot together with the profiles from the presently active glider data file. The choices are temperature or salinity profiles, or both, from either the GDEM monthly global climatology, from the MOODS global profile archive, or from nearby glider observations from the same glider that is presently being displayed.

Activating the **GDEM** radiobuttons cause the nearest (in position and time of year) monthly mean profile to be drawn as a thick dashed cyan-colored line. In addition the two profiles equal to the GDEM mean profile plus and minus the specified number of standard deviations (of temperature or salinity at that position and time of year versus depth) are also drawn as thick dashed cyan-colored lines. The number of standard deviations (default is three) can be changed by entering the desired value in the edit boxes adjacent to the **GDEM** temperature and salinity radiobuttons.

Temperature or salinity profiles from the MOODS archive are drawn on the profile plot as thin grey lines when the **MOODS Archive** radiobuttons are activated. Activation displays all MOODS profiles within 50 km of the glider position and within 30 days (time of year) of the glider observation date. If fewer than 50 profiles are retrieved within this space and time domain from MOODS, then the search distance is increased by 50 km as many times as is required to obtain at least 50 profiles.



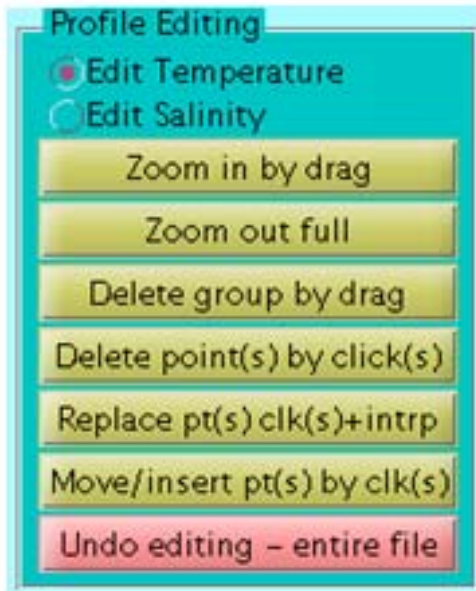
**Figure 23** The "profile underlays" group of uicontrols.

Activating the **Glider T** and **Glider S** radiobuttons causes recent (and future) profiles measured by the presently displayed glider to be drawn as thin grey lines. All available profiles available within the specified time window are drawn. The default time window is three days, but can be changed by entering the desired number of days into the edit box next to the **Glider T** and **Glider S** radiobuttons. Fractional days may be entered.

## 7.6 Editing temperature and salinity profiles

Temperature and salinity profile values are changed or flagged using pushbuttons and radiobuttons in the "profile editing" group of uicontrols (Figure 13). A closely related

topic, displaying and changing QC flags and flagged values set during automated QC are discussed in the next (in Section 7.7).



**Figure 24** The "profile editing" group of uicontrols.

The present version of LAGER allows only temperature and salinity to be edited and flagged. In order to edit temperature, the **Edit Temperature** radiobutton must be on. Similarly, the **Edit Salinity** radiobutton must be on to edit salinity. The ascending and descending profiles must be edited separately. Therefore, if the presently active data file has both an ascending and descending profile, one or the other must be turned off (not displayed) by setting one of the following radiobuttons in the "profile display" group of uicontrols: **Show Ascending**, **Show Descending**, **Show Each**.

The profile plot can be expanded to focus on a particular depth range by pressing the **Zoom in by drag** pushbutton. After pushing the pushbutton, left click and hold down the mouse button at one end of the desired depth range. Then, while continuing to hold the mouse button down, drag the mouse to the other end of the desired depth range and then release. Only the vertical coordinates defined by dragging the mouse are used in the zooming, not the horizontal coordinates. When the plot is redrawn in the new zoomed mode, the horizontal axes for each profile variable are rescaled to fit the plot. To zoom back to the original profile plot vertical axis, press the **Zoom out full** pushbutton.

A set of points can be deleted by first pressing the **Delete group by drag** pushbutton, but make sure that the correct radiobutton (**Edit Temperature** or **Edit Salinity**) is on first. Next, left click the mouse and hold down at one corner of a box that will contain the set of points to be deleted. Then, while holding the mouse button down, drag the mouse to the other diagonal corner of the box. When the mouse button is released, all observation points (for the variable selected – either temperature or salinity) within the box will

disappear from the plot. The manual QC flag for these points is now set to indicate a bad value (manual\_temp\_flag = 3 or manual\_salt\_flag = 3, see Sections 5.2.1 and 5.2.2).

Individual points can also be deleted (marked as bad by setting the manual temperature or salinity flag to 3). First press the **Delete point(s) by clicks(s)** button. Next left click the mouse on the point to be deleted. Continue deleting more points by left clicking on the points to be deleted. At any time, the previous deleted points can be undeleted by clicking the center mouse button (at any position within the plot axes). Center click once for each point to be undeleted (which will be done in reverse order of their deletion). When finished deleting (or undeleting), press the right button of the mouse to finish. Once finished, the deleted points cannot be undeleted.

Note that points that have been marked bad either by the automatic QC routines or by manual editing within MUG will not be shown on the plot and will leave a blank gap in the profile plot. However, this behavior also depends upon the settings in the "flags and flagged data" group of uicontrols, discussed in the next section. Controls in that group can be set to display data marked as bad and optionally to draw horizontal lines through points that have been marked as bad.

Data gaps can be filled by use of the **Replace pt(s) clk(s)+interp** pushbutton. A data gap is defined here as an uninterrupted series of points that are either missing (filled with the missing value indicator) or that have been marked as bad either by the automated QC routines or by manual editing. To fill a data gap, press the **Replace pt(s) clk(s)+interp** pushbutton. Then left click within the depth gap. The bad and missing points will be linearly interpolated from the first good (or not missing) points at the ends of the gap. Continue filling other gaps by clicking the left mouse button in the center of the gaps. At any time, the center button can be pressed to undo the previous interpolation. Continue to press the center button to undo as many gap interpolations as desired. Press the right mouse button to finish the interpolations and undo-interpolations.

Individual point values in either temperature or salinity (depending upon the settings of the **Edit Temperature** and **Edit Salinity** radiobuttons) can be moved (changed to another value) using the **Move/insert pt(s) by clk(s)** pushbutton. However, the depth of a point value cannot be changed. First press the **Move/insert pt(s) by clk(s)** pushbutton. Then left click next to a point value at the position of the new value. The plot will immediately be redrawn with the point moved to the mouse click position. Continue moving that same point or any other point by left clicking. At any time the center button can be click to undo previous moves in the reverse order that they were moved. To finish moving and unmoving, press the right mouse button.

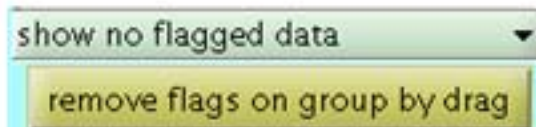
Sometime the operator might wish to remove various edits to a file in MUG. The entire active data file can be returned back to its original state prior to manual editing by pressing the **Undo editing – entire file** pushbutton. This button is colored red to help prevent the user from accidentally hitting it because all previous manual edits to that file will be lost (for both the descending and ascending profiles).

## 7.7 Changing and displaying flags

The "flags and flagged data" group of uicontrols (Figure 14) together with **Flag Definition** pushbutton (Figure 15) allow flags to be displayed in the profile plot, removed, and described. To change the flag display or flagged data display behavior on the profile plot, press the **show no flagged data** popup menu button. The menu will then show four lines:

- show no flagged data
- show all flagged data
- sho no flagged data/show flags
- sho flagged data/show flags.

Select the desired type of display for the profile plot by selecting one of the options in the popup menu. No data flagged as bad either by the automated QC routines or by manual QC is plotted if "show no flagged data" is selected (lines 1 and 3), but they are shown if "show all flagged data" or "sho flagged data" (lines 2 and 4) are selected. If lines 3 or 4 are selected then horizontal lines are plotted across the entire width of the profile plot at each depth where a value is flagged as bad. Temperature flag lines are black and salinity flag lines are red. The salinity flag lines are drawn last, so they may cover a black line if both temperature and salinity are bad at the same depth.



**Figure 25** The "flags and flagged data" group of uicontrols.

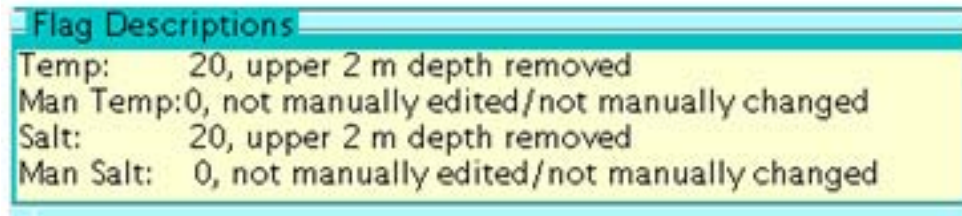
Flags marking a series of points (versus depth) for either temperature or salinity (depending upon how the Edit Temperature or Edit Salinity radiobuttons are set) can be removed by use of the **remove flags on group by drag** pushbutton. Press this pushbutton then left click the mouse on the profile at the starting depth of points where the bad flags are to be removed. Continue hold the left mouse button down and draw the mouse to the bottom depth of the group of points where the bad flags are to be removed. When the mouse button is released, the automated flags and manual flags (temp\_flag and temp\_manual\_flag or salt\_flag and salt\_manual\_flag) are set to zero (indicating a good values).



**Figure 26** The Flag Definition pushbutton.

A description of temperature and salinity automated and manual flags set at any depth can be obtained by first pressing the **Click on Flag Line for Definition** pushbutton. Then left click at any depth in the profile plot. After clicking, the Click on Flag Line for Definition pushbutton is replaced by the four-line description of the flags at that depth (example shown in Figure 16). Continue left clicking the mouse at other depths to obtain their flag descriptions. When finished, press the right mouse button. Typically, the mouse is clicked at depth showing the horizontal lines that indicate that the temperature

or salinity is marked as bad (after setting the **show no flagged data** popup menu to either line 3 or 4).



**Figure 27** Flag description text box describing temperature and salinity flags. These lines describe the flags at the depth pointed to on the profile plot by left clicking.

## 7.8 Exit, Quit, and generating BUFR file

As each file is viewed by cycling through the selected files, the file is renamed in the glider data archive directories by changing the file ending to `_editmanf.nc`. The "f" indicates to the LAGER software that it has gone through manual editing. Files at this state are ready to reformat into BUFR files and to pass on to the RTDHS. The user has three options related to suspending the present MUG editing session and sending finished files on to BUFR and RTDHS.

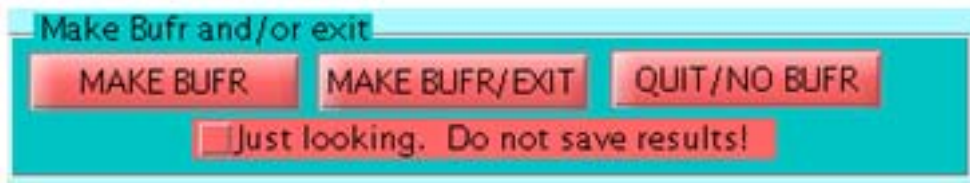
By pressing the **MAKE BUFR** pushbutton, all files ending in `_editmanf.nc` anywhere throughout the entire glider data archive are reformatted into BUFR files. The BUFR files are then moved to the directory defined by `setup.going_to_rtdhs_dir` in the `glider_qc_info.m` file (see Appendix B), which is the pick up directory for files going to the RTDHS. Then, each of these glider data files is renamed by replacing the ending with `_edit.nc`. The MUG editing session is then continued. However, since the set of files meeting the specification requested in the "initial setup" group of uicontrols has changed, the entire session is essentially restarted.

The second option, implemented by pressing the **MAKE BUFR/EXIT** pushbutton, is the same as the first except that once the BUFR files are made and the data files are renamed, the MUG session is terminated.

The third option is to terminate the MUG editing session without creation of the BUFR files and without renaming the `_editmanf.nc` files. When the user starts up MUG for another session, the files edited so far (and now ending in `_editmanf.nc`) can be examined again and changed if necessary. Also, by changing the settings of the three radiobuttons in the "initial setup" group of uicontrols (**editmanf**, **editman**, **edit**), any subset of file types (depending upon file ending) can be brought in for editing.

The final button in this group is the **Just looking. Do not save results!** toggle button. After this switch is turned on, either at the beginning of the editing session or at any time after, none of the edits or changes to the files are written back to the file and the file name

is not changed to `_editmanf.nc` (although the file may already have this ending). This setting is useful for browsing through the profile data set without making any changes.



**Figure 28** The "quit, exit, BUFR" group of uicontrols.

## 8 APPENDIX A

Lager subdirectories within the main LAGER directory and software files contained within them are shown below. Directory names are written as all bold-font capital letters and software file names are displayed in italicized lower case ending in the file extension (e.g., .m, .f). Sub functions (MATLAB) or subroutines (Fortran) contained within a software file are listed indented below the software file name.

**LAGER** (base directory for all software)

**AUTOQC** (automated quality control)

*autoqc2.m* (performs automated quality control)

*getgdem*

*ctd\_conductivity\_correction*

*ctd\_conductivity\_correction\_slocum*

*stabilize\_prof*

*check\_static\_stability*

*divide\_profiles*

*check\_whole\_profile\_flags*

*do\_gradient\_test*

*do\_spike\_test*

*correct\_conductivity.m*

*dcdt.m*

*perform\_auvb\_calibration.m*

*perform\_calibration.m*

*plot\_qc.m*

**BATHYMETRY** (not implemented yet, but will hold regional high-resolution bottom bathymetry files).

**BUFR**

*glider\_nc\_to\_buf.f* (generates bufr format files from qc'd netcdf glider files).

*read\_ncdump*

*replacechars*

*getrvariable*

*getidvariable*

*getivariable*

*datevec*

*datenum*  
*glider\_nc\_to\_buf.r.com* (csh file called from MATLAB  
to start *glider\_nc\_to\_buf.r.f*).  
*decode\_buf.r.f* (dumps buf.r file contents to an  
ascii file).  
*buf.r\_to\_ascii.m* (interprets buf.r dump produced by  
*decode\_buf.r.f* and writes readable text file).  
*decode\_buf.r\_MATLAB.com* (csh command file that  
calls *decode\_buf.r.f* and *buf.r\_to\_ascii.m*).  
*decode\_buf.r.com.m* (MATLAB program to decode all  
Buf.r files in a directory).  
*Makefile* (makefile to compile and link  
*decode\_buf.r.f* and *glider\_nc\_to\_buf.r.f*).

## **CRON**

*glider\_import\_qc.com* (csh script executed by CRON  
start automated LAGER import and QC  
analysis).  
*glider\_import\_qc.cron* (cron script).  
*README* (short document describing how to set up  
and run a cron job).

## **DOCS**

### **IMPORT\_SEAGLIDER**

*import\_seaglider.m*  
*update\_seaglider\_version.m*

### **IMPORT\_SLOCUM**

*import\_slocum.m*  
*fixgaps.m*  
*isolated.m*  
*myrunningave.m*  
*read\_sbd.m*

## **INFOFILES**

*Areas*  
*bam.m*  
*bam.p*  
*calibration\_tables.m*  
*glider\_qc\_info.m* (main LAGER directory set up  
routine).

*Newnames*

### **devs**

*auvb\_007.bam*  
*auvb\_008.bam*  
*auvb\_012.bam*  
*auvb\_013.bam*  
*auvb\_014.bam*

## **MANUALQC**

*mug.m*  
*mug.fig*  
*plotprof\_multvars.m*  
*readmoodsrandacc\_init.m*  
*readmoodsrandacc.m*

## OTHERFORMATS

*compare\_lager\_to\_m2k.m*  
*moods2mat.m*

## SCRIPTS

*glider\_import\_qc.m* (main LAGER automated glider  
import and analysis script).  
*get\_areas.m*  
*get\_newnames.m*  
*make\_buf\_r\_file.m*  
*rename\_sbd.m*

## SLOCUM\_BASE

*dba2\_orig\_MATLAB* (C executable)  
*dba\_sensor\_filter* (C executable)  
*dbd2asc* (C executable)  
*rename\_dbd\_files* (C executable)  
*mysensors.txt* (used with *dba\_sensor\_filter*)  
**glider\_cache**  
*10d68961.cac* (one example).

## UTILITIES

*compare\_to\_rutgers.m*  
*plot\_glider\_pos\_map.m*  
*plot\_vert\_sect.m*  
*slocum\_to\_nc.m*  
*slocum\_variables\_stats.m*  
*vert\_ave\_currents.m*

# 9 APPENDIX B

The following is a listing from the *glider\_qc\_info.m* file.

```

function setup=glider_qc_info;
%
%  program glider_qc_info.m
%
setup.incoming_seaglider_dir='/projects/isop/GLIDERQC/INCOMING_SEAGLIDER';
setup.incoming_slocum_dir='/projects/isop/GLIDERQC/INCOMING_SLOCUM';
setup.going_to_rtdhs_dir='/projects/isop/GLIDERQC/GOING_TO_RTDHS';
setup.slocum_executables_directory='/projects/isop/GLIDERQC/SLOCUM_BASE';
setup.buf_r_executables_directory='/projects/isop/GLIDERQC/BUFR';
setup.glider_cache='/projects/isop/GLIDERQC/SLOCUM_BASE/glider_cache';
%  very important!! the buf_r_tables_directory definition must end in a /
%  e.g., = '/projects/isop/BUFR/LIB/bufrtables/';
setup.buf_r_tables_directory='/projects/isop/BUFR/LIB/bufrtables/';
setup.plot_directory='/projects/isop/GLIDERQC/PLOTS';
%
%  navo_method = 0 no, not at navo
%               = 1 all profiles will be manually edited, so the autoqc
%               program will produce files ending in _editman.nc,
%               indicating that it needs to be manually edited.

```

```

%           this also has lots of ramifications for the mug manual
%           editor.  navo_method > 0 also means that the import
%           of data from the incoming slocum and seaglider files
%           is handled differently than when navo_method=0.
%
setup.navo_method=1;
%
%
%   the base_qc_data_directory contains the glider project data subdirectories
%
setup.base_qc_data_directory='/home/autoglider/GOC_data';
%
%   temporary directory
%
setup.temporary_directory='/tmp';
%
%   directory containing the GDEM monthly temperature, salinity,
%   temperature standard deviation, and salinity standard deviation
%   netcdf files.
%
setup.gdem_directory='/projects/isop/CLIMS/GDEMv3';
%
%   directory containing the binary version of the MOODS profiles interpolated
%   to standard (GDEMv3) depths.
%
setup.moods_directory='/scratch/MAW2';
%
%   the start and stop times of the data to be qc'd by the mug editor
%   are relative to today.  The units of start_days and stop_days is days.
%   The mug gui allows these times (starting and ending dates - not
%   relative days) to be modified at the start of an editing session.
%
setup.start_days=-3;
setup.stop_days=1;% a little into the future is good to avoid GMT problems.
%
%   group and permissions
%
setup.group = 'autoglider';
setup.permissions= '2770';
%

```

## 10 APPENDIX C

The following is an ncdump of a version 2.f (newest version) Seaglider NetCDF raw file produced by BaseStation. This list shows only the subset of variables that are a function of time. All variables are of type "float" (4-byte real numbers) except time which is of type "int" (4-byte integer). The missing value for each variable is nan (not a number).

```

float eng_elaps_t_0000(time) ;
    eng_elaps_t_0000:units = "seconds" ;
    eng_elaps_t_0000:standard_name = "time" ;
    eng_elaps_t_0000:description = "seconds since start of mission" ;
float eng_elaps_t(time) ;
    eng_elaps_t:units = "seconds" ;
    eng_elaps_t:standard_name = "time" ;
    eng_elaps_t:description = "seconds since start of dive" ;
float eng_condFreq(time) ;
float eng_tempFreq(time) ;
float eng_depth(time) ;
    eng_depth:units = "cm" ;
    eng_depth:standard_name = "depth" ;
    eng_depth:description = "vertical distance below the surface" ;
float eng_head(time) ;
float eng_pitchAng(time) ;

```

```

float eng_rollAng(time) ;
float eng_pitchCtl(time) ;
float eng_rollCtl(time) ;
float eng_vbdCC(time) ;
float wlbb2f_redRef(time) ;
float wlbb2f_redCount(time) ;
float wlbb2f_blueRef(time) ;
float wlbb2f_blueCount(time) ;
float wlbb2f_fluorCount(time) ;
float wlbb2f_VFtemp(time) ;
int time(time) ;
    time:missing_value = nan ;
    time:units = "seconds" ;
    time:standard_name = "time" ;
    time:description = "sample time in GMT epoch format" ;
    time:reference = "00:00Z 1 January 1970" ;
float depth(time) ;
    depth:missing_value = nan ;
    depth:units = "m" ;
    depth:standard_name = "depth" ;
    depth:description = "distance below the surface" ;
float latitude(time) ;
    latitude:missing_value = nan ;
    latitude:units = "Decimal degrees" ;
    latitude:standard_name = "lat" ;
    latitude:description = "sample latitude" ;
float longitude(time) ;
    longitude:missing_value = nan ;
    longitude:units = "Decimal degrees" ;
    longitude:standard_name = "lon" ;
    longitude:description = "sample longitude" ;
float pressure(time) ;
    pressure:missing_value = nan ;
    pressure:units = "dbar" ;
    pressure:standard_name = "sea_water_pressure" ;
float temp(time) ;
    temp:missing_value = nan ;
    temp:units = "C" ;
    temp:standard_name = "temperature (in situ)" ;
    temp:description = "corrected for first-order lag" ;
float conductivity(time) ;
    conductivity:missing_value = nan ;
    conductivity:units = "S m^-1" ;
    conductivity:standard_name = "electrical_conductivity" ;
    conductivity:description = "corrected for first-order lag" ;
float salinity(time) ;
    salinity:missing_value = nan ;
    salinity:units = "none" ;
    salinity:standard_name = "salinity" ;
float sigma_t(time) ;
    sigma_t:missing_value = nan ;
    sigma_t:units = "kg/m^3" ;
    sigma_t:standard_name = "potential density" ;
    sigma_t:ref_pressure = "0" ;
float theta(time) ;
    theta:missing_value = nan ;
    theta:units = "C" ;

```

```

        theta:standard_name = "potential temperature" ;
        theta:description = "corrected for first-order lag" ;
float density(time) ;
    density:missing_value = nan ;
    density:units = "kg/m^3" ;
    density:standard_name = "density (in situ)" ;
float sigma_theta(time) ;
    sigma_theta:missing_value = nan ;
    sigma_theta:units = "kg/m^3" ;
    sigma_theta:standard_name = "potential density" ;
    sigma_theta:ref_pressure = "0" ;
float horiz_speed_pitch_w(time) ;
    horiz_speed_pitch_w:missing_value = nan ;
    horiz_speed_pitch_w:units = "cm/s" ;
    horiz_speed_pitch_w:standard_name = "Horizontal speed, from
        observed pitch and vertical velocity" ;
float horz_speed_pitch_buoy_model(time) ;
    horz_speed_pitch_buoy_model:missing_value = nan ;
    horz_speed_pitch_buoy_model:units = "cm/s" ;
    horz_speed_pitch_buoy_model:standard_name = "Horizontal speed
        from model (buoy and pitch)" ;
float vert_speed_pitch_buoy_model(time) ;
    vert_speed_pitch_buoy_model:missing_value = nan ;
    vert_speed_pitch_buoy_model:units = "cm/s" ;
    vert_speed_pitch_buoy_model:standard_name = "Vertical speed from
        model (buoy and pitch)" ;
float glide_angle_model(time) ;
    glide_angle_model:missing_value = nan ;
    glide_angle_model:units = "cm/s" ;
    glide_angle_model:standard_name = "Glide angle from model (buoy
        and pitch)" ;
float north_displacement_model(time) ;
    north_displacement_model:missing_value = nan ;
    north_displacement_model:units = "m" ;
    north_displacement_model:standard_name = "Northward displacement
        from model" ;
float east_displacement_model(time) ;
    east_displacement_model:missing_value = nan ;
    east_displacement_model:units = "m" ;
    east_displacement_model:standard_name = "Eastward displacement
        from model" ;
float Buoyancy(time) ;
    Buoyancy:missing_value = nan ;
float u_da ;
    u_da:units = "m/s" ;
    u_da:standard_name = "depth average current" ;
    u_da:description = "Eastward component of depth averaged current";
float v_da ;
    v_da:units = "m/s" ;
    v_da:standard_name = "depth average current" ;
    v_da:description = "Northward component of depth averaged
        current" ;

```

## 11 APPENDIX D

The following is an ncdump of the old version (used prior to July 2008) Seaglider NetCDF raw file produced by BaseStation. This list shows only the subset of variables that are a function of time. All variables are of type "float" (4-byte real numbers) except time which is of type "int" (4-byte integer). The missing value for each variable is inf (not a number).

```
float elaps_t_0000(time) ;
    elaps_t_0000:standard_name = "time" ;
    elaps_t_0000:units = "seconds" ;
    elaps_t_0000:description = "seconds since start of mission" ;
float elaps_t(time) ;
    elaps_t:standard_name = "time" ;
    elaps_t:units = "seconds" ;
    elaps_t:description = "seconds since start of dive" ;
float condFreq(time) ;
float tempFreq(time) ;
float depth(time) ;
    depth:standard_name = "depth" ;
    depth:units = "cm" ;
    depth:description = "vertical distance below the surface" ;
float head(time) ;
float pitchAng(time) ;
float rollAng(time) ;
float pitchCtl(time) ;
float rollCtl(time) ;
float vbdCC(time) ;
float redRef(time) ;
float redCount(time) ;
float blueRef(time) ;
float blueCount(time) ;
float fluorCount(time) ;
float VFtemp(time) ;
int time(time) ;
    time:missing_value = 0 ;
    time:standard_name = "time" ;
    time:units = "seconds" ;
    time:description = "time of sample in GMT epoch format" ;
float depth_m(time) ;
    depth_m:missing_value = inf ;
    depth_m:standard_name = "depth" ;
    depth_m:units = "m" ;
    depth_m:description = "vertical distance below the surface" ;
float lat(time) ;
    lat:missing_value = inf ;
    lat:standard_name = "lat" ;
    lat:units = "decimal degrees" ;
    lat:description = "estimated latitude" ;
float lon(time) ;
    lon:missing_value = inf ;
    lon:standard_name = "lon" ;
    lon:units = "decimal degrees" ;
    lon:description = "estimated longitude" ;
float Pressure(time) ;
    Pressure:missing_value = inf ;
```

```

        Pressure:standard_name = "sea_water_pressure" ;
        Pressure:units = "dbar" ;
float TempC_Cor(time) ;
        TempC_Cor:missing_value = inf ;
        TempC_Cor:standard_name = "sea_water_temperature" ;
        TempC_Cor:units = "C" ;
        TempC_Cor:description = "corrected for first-order lag" ;
float Cond_Cor(time) ;
        Cond_Cor:missing_value = inf ;
        Cond_Cor:standard_name = "sea_water_electrical_conductivity" ;
        Cond_Cor:units = "S m-1" ;
        Cond_Cor:description = "corrected for first-order lag" ;
float Salinity(time) ;
        Salinity:missing_value = inf ;
        Salinity:standard_name = "sea_water_salinity" ;
        Salinity:units = "1e-3" ;
float SigmaT(time) ;
        SigmaT:missing_value = inf ;
float TempC_Cor_pot(time) ;
        TempC_Cor_pot:missing_value = inf ;
        TempC_Cor_pot:standard_name = "potential sea_water_temperature" ;
        TempC_Cor_pot:units = "C" ;
        TempC_Cor_pot:description = "corrected for first-order lag" ;
float Density(time) ;
        Density:missing_value = inf ;
        Density:standard_name = "seawater density" ;
float Density_pot(time) ;
        Density_pot:missing_value = inf ;
        Density_pot:standard_name = "potential seawater density" ;
float horiz_speed_pitch_w(time) ;
        horiz_speed_pitch_w:missing_value = inf ;
        horiz_speed_pitch_w:standard_name = "Horizontal speed, from
observed pitch and vertical velocity" ;
        horiz_speed_pitch_w:units = "cm/s" ;
float horiz_speed_pitch_buoy_model(time) ;
        horiz_speed_pitch_buoy_model:missing_value = inf ;
        horiz_speed_pitch_buoy_model:standard_name = "Horizontal speed
from model (buoy and pitch)" ;
        horiz_speed_pitch_buoy_model:units = "cm/s" ;
float vert_speed_pitch_buoy_model(time) ;
        vert_speed_pitch_buoy_model:missing_value = inf ;
        vert_speed_pitch_buoy_model:standard_name = "Vertical speed from
model (buoy and pitch)" ;
        vert_speed_pitch_buoy_model:units = "cm/s" ;
float glide_angle_model(time) ;
        glide_angle_model:missing_value = inf ;
        glide_angle_model:standard_name = "Glide angle from model (buoy
and pitch)" ;
        glide_angle_model:units = "cm/s" ;
float north_displacement_model(time) ;
        north_displacement_model:missing_value = inf ;
        north_displacement_model:standard_name = "Northward displacement
from model" ;
        north_displacement_model:units = "m" ;
float east_displacement_model(time) ;
        east_displacement_model:missing_value = inf ;

```

```

    east_displacement_model:standard_name = "Eastward displacement
from model" ;
    east_displacement_model:units = "m" ;
float Buoyancy(time) ;
    Buoyancy:missing_value = inf ;

```

## 12 REFERENCES

Boyer, T. and S. Levitus, 1994: Quality control processing of historical oceanographic temperature, salinity, and oxygen data, National Oceanographic Data Center, Ocean Climate Laboratory.

C-T. Chen, C. -T. and F.J Millero, 1977: Speed of sound in seawater at high pressures. J. Acoust. Soc. Am, 62, 1129-1135.

Eriksen, C.C., T.J. Osse, R.D. Light, T. Wen, T.W. Lehman, P.L. Sabin, J.W. Ballard, and A.M. Chiodi, 2001: Seaglider: A long-range autonomous underwater vehicle for oceanographic research, IEEE Journal of Oceanic Engineering, 26 (4), 424-436.

Gronell, A. and S. E. Wijffels, 2008: A semiautomated approach for quality controlling large historical ocean temperature archives, J. Atmos. and Ocean. Tech., 25, 990-1003.

Ingleby, B. and M. Huddleston, 2007: Quality control of ocean temperature and salinity profiles – Historical and real-time data, J. Mar. Systems, 65, 158-175.

Jackett, D.R., and T.J. McDougall, 1995: Minimal adjustment of hydrographic profiles to achieve static stability, J. Atmos. Ocean. Tech., 12, 381-389.

Johnson, G. C., J. M. Toole, and N. G. Larson, 2007: Sensor corrections for Sea-Bird SBD-41CP and SBE-41 CTDs, J. Atmos. Ocean. Tech, 24, 1117-1130.

Kerfoot, J, S Glenn, J. Kohut, O. Schofield, H. Roarty: Correction for sensor mismatch and thermal lag effects in non-pumped conductivity-temperature sensors on the Slocum coastal electric glider, Ocean Sciences, Honolulu, Hawaii, February 2006.

Levitus, S., 2005, World Ocean Database 2005 documentation, NODC Internal Report 18, U.S. Government Printing Office, Washington D.C., 163 pp.

Lueck, R. G., 1990: Thermal inertia of conductivity cells: Theory. J. Atmos. Ocean. Tech., 7, 741-755.

Lueck, R. G., 1990: Thermal inertia of conductivity cells: observations with a Sea-Bird Cell. J. Atmos. Ocean. Tech., 7, 756-768.

Maudire, G., 1994: Routine data quality control in a data centre. The example of the TOGA/WOCE Subsurface Centre, OCEANS '94, "Oceans Engineering for Today's Technology and Tomorrow's Preservation." Proceedings, Vol 2, II/384-II389.

- McDougall, T.J., 1987: Neutral surfaces, *J. Phys. Ocean.*, 17, 1950-1964.
- McDougall, T.J. and D.R. Jackett, 1995: Minimal adjustment of hydrographic profiles to achieve static stability, *J. Atmos. Ocean. Tech.*, 12, 381-389.
- Millero, F. J., & Li, X. (1994). Comments on “On equations for the speed of sound in seawater” [*J. Acoust. Soc. Am.* 94, 255-275 (1993)]. *The Journal of the Acoustical Society of America*, 95, 2757–2759.
- Morison et al., 1994: The correction for thermal-lag effects in Sea-Bird CTD data. *J. Atmos. Ocean. Tech.*, 11, 1151-1164.
- Schmid, C., R.L. Molinari, R. Sabina, Y.-H. Daneshzadeh, X. Xia, E. Forteza, and H. Yang, 2007: The real-time data management system for Argo profile float observations, *J. Atmos. Ocean. Tech.*, 24, 1608-1628.
- Schofield, O., J. Kohut, D. Aragon, L. Creed, J. Graver, J. Kerfoot, H. Roarty, C. Jones, D. Webb, S. Glenn, 2007: Slocum Gliders: Robust and Ready, *Journal of Field Robotics*, 24(6), 473-485.
- Sea-Bird, 2008: Glider Payload CTD, Sea-Bird Electronics, Inc., Preliminary Brochure, [www.seabird.com/products/GliderCTDs.htm](http://www.seabird.com/products/GliderCTDs.htm)
- Sherman, J.T., R.E. Davis, W.B. Owens, and J. Valdes, 2001: The autonomous underwater glider 'Spray.' *IEEE Oceanic Eng.*, 26, 437-446.
- UNESCO, 1990: GTSP Real-Time Quality Control Manual, Intergovernmental Oceanographic Commission, Manual and Guides 22, SC/90/WS-74, 121 pp.